TRANSFERENCIA DE APRENDIZAJE APLICADO A RECONOCIMIENTO DE PATRONES EN IMÁGENES MÉDICAS

TESIS DE MAESTRÍA EN INGENIERÍA

Camilo Plata
Alumno

Damián G. Hernández
Director

Inés Samengo
Codirectora

Instituto Balseiro
Centro Atómico Bariloche
Junio 2020

Resumen

En el presente trabajo se aplicó la técnica de transferencia de aprendizaje a la tarea de clasificación de imágenes de fondo de ojo, para la detección de retinopatía diabética utilizando un set de datos en el rango de muy pocas imágenes. La transferencia de aprendizaje se realizó utilizando las redes Xception, Inception V3 y DenseNet 169, preentrenadas con la base de datos ImageNet. Se realizó un finetuning de los pesos, utilizando las técnicas de dropout, regularización L2 y data augmentation para mejorar la generalización durante el entrenamiento. Se emplearon las técnicas CAM y Guided Grad CAM con el fin de observar las regiones de mayor importancia en las imágenes a la hora de realizar una clasificación. Se corroboró que los criterios aplicados por las soluciones encontradas son apropiados para este problema. Se logró una precisión de 76 % empleando imágenes de 224x224 pixeles, y una precisión de 84 % con imágenes de 448x448. A la salida de la anteúltima capa de red y se aplicó el método de reducción de dimensionalidad t-SNE, observando una relación entre formación de clusters en un espacio de pocas dimensiones y un patrón diferenciable para cada categoría en la salida. Finalmente, se calculó el área bajo la curva ROC para una base de datos de testeo, obteniendo un valor de 0.91.

Índice general

Capítulo 1

Introducción

1.1. Motivación

La retinopatía diabética es una patología causada por el debilitamiento de los vasos sanguíneos de la retina por los altos niveles de glucosa en sangre. El método usual de detección y clasificación de la enfermedad en sus distintas etapas se realiza a través del análisis, por un especialista en retina, de imágenes de fondo de ojo. Cerca de un tercio de las personas con diabetes padecen esta enfermedad y el daño causado en etapas avanzadas de la misma puede llevar a la pérdida de la visión, siendo la primera causa de ceguera en la población adulta [1]. Para muchos de los pacientes, la detección de la enfermedad se realiza de forma tardía, en etapas cuyos efectos son irreversibles, por lo que es importante el desarrollo de técnicas para ayudar a una detección temprana. En este marco, algoritmos de inteligencia artificial pueden ofrecer una solución con resultados que podrían cambiar sustancialmente el panorama.

En los últimos años, dichos algoritmos han tenido éxito resolviendo tareas en distintas áreas, tan diversas como reconocimiento de objetos, detección de fraude, reconocimiento de voz, hasta asistencia a médicos en el área de la

salud. Un inconveniente de estos algoritmos es que necesitan gran cantidad de datos para alcanzar un buen desempeño. Particularmente en nuestra área de interés, la clasificación de imágenes médicas, es frecuente encontrarse con datasets de imágenes muy chicos como para entrenar redes convolucionales profundas, un tipo de algoritmo empleado en el campo de detección de objetos. Una de las estrategias propuestas para sortear este problema, es el empleo de redes convolucionales que han sido entrenadas previamente con otra base de dato, utilizando millones de imágenes y para la clasificación de otro tipo de targets, transfiriendo posteriormente parte de la información aprendida a la nueva tarea en cuestión. Dicha técnica, denominada transferencia de aprendizaje, se ha empleado en distintos campos con resultados prometedores. En el presente trabajo se utiliza la técnica de transferencia de aprendizaje aplicada a la tarea de clasificación de imágenes de fondo de ojo para detección de retinopatía diabética, trabajando en el rango de pocas imágenes, utilizando distintas redes preentrenadas y evaluando su desempeño con distintos métodos.

1.2. Objetivos

- Utilizar la técnica de transferencia de aprendizaje en el rango de pocas imágenes, para aprender la tarea de clasificación de imágenes de fondo de ojo, detectando retinopatía diabética.
- Optimizar hiperparámetros, empleando distintas técnicas para mejorar la generalización en el proceso de aprendizaje.
- Utilizar técnicas que permitan entender y visualizar el procesamiento de información en dichas redes, particularmente que permitán comprender criterios aprendidos por la red para realizar la clasificación.

1.3. Retinopatía diabética

La retinopatía diabética es una complicación originada por una alteración en los vasos sanguíneos de la retina, dañando este tejido ocular y siendo la principal causa de ceguera de personas en edad laboral. Actualmente, se incita constantemente al desarrollo técnicas o herramientas que ayuden a identificar pacientes potenciales, ya que es difícil detectar y prevenir esta enfermedad, pues se require que el diagnóstico sea hecho por un médico especialista.

1.3.1. Etapas de la enfermedad

La enfermedad avanza gradualmente y se indentifican distintas etapas dentro de la misma, las cuales van desde la retinopatía leve no proliferativa hasta el estadío proliferativo, siendo esta última etapa la que más complicaciones tiene, con un alto riesgo de ceguera para el paciente. Las distintas etapas de la enfermedad se encuentran listadas en la tabla 1.1, indicando además las características observables en cada una. Las escalas de severidad presentadas fueron consensuadas por la Clínica Internacional de Retinopatía Diabética (ICDR), correspondientes a categorías que van desde 0 para personas sanas, llegando hasta la categoría 4, siendo la más severa y con más riesgos para la visión del paciente.

Tabla 1.1: Etapas de la retinopatía diabética.

Categoría	Grado de avance	Características observables
0	Sin Retinopatía	Sin anomalías
1	Leve	Microaneurismas y microhemorragias
2	Moderada	Microaneurismas y exudados duros
3	Severa	Alguno de los siguientes: >=20 microaneurismas en cada uno de los 4 cuadrantes, vasos sanguíneos irregulares en >=2 cuadrantes, abnormalidades microvasculares intraretinales en >=1 cuadrante
4	Proliferativa	neovascularización y/o hemorragias intraretinales o vitreas

En la figura 1.1 se muestra una imagen de fondo de ojo, donde se resaltan microaneurismas. Se observan como puntos rojos oscuros, los cuales pueden tener distintos tamaños y aparecer en cualquier lugar dentro de la retina. Son en general el primer signo visible de retinopatía diabética en los pacientes, encontrándose por lo tanto en la étapa 1 de la misma.

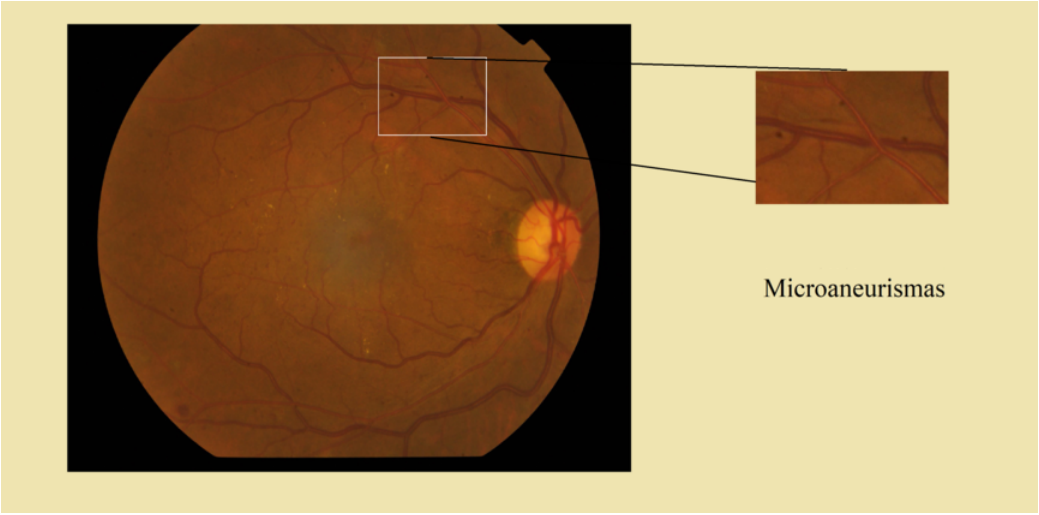


Figura 1.1: Imagen de fondo de ojo con microaneurismas.

En la imágen 1.2 se muestran exudados duros, lesiones que aparecen en las imágenes de fondo de ojo debido a depósitos lipídicos. La caraterística distintiva de este tipo de lesiones es la de tener bordes definidos y generalmente aparecen alrededor de la mácula, la zona más oscura ubicada en el centro. Pacientes con este tipo de lesiones se encuentran en la etapa 2 de la enfermedad.

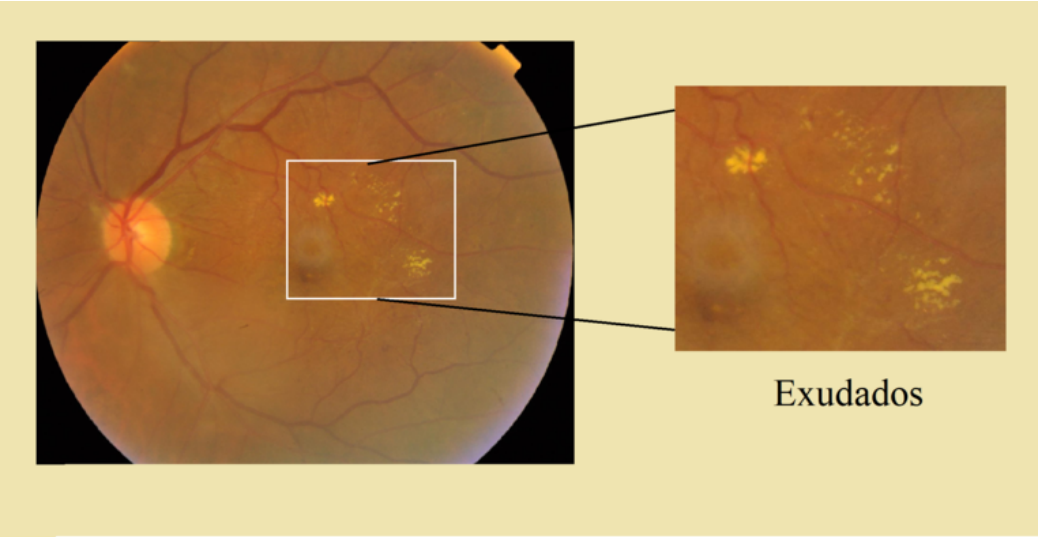


Figura 1.2: Imagen de fondo de ojo con exudados duros.

En la figura 1.3 se observan resaltados los llamados cotton wool spots o exudados algodonosos, los cuales tienen mayor tamaño que los exudados duros y tienen bordes menos definidos. Son causados por isquemias locales en la zona, es decir, disminución de la circulación de sangre en las regiones adyacentes.

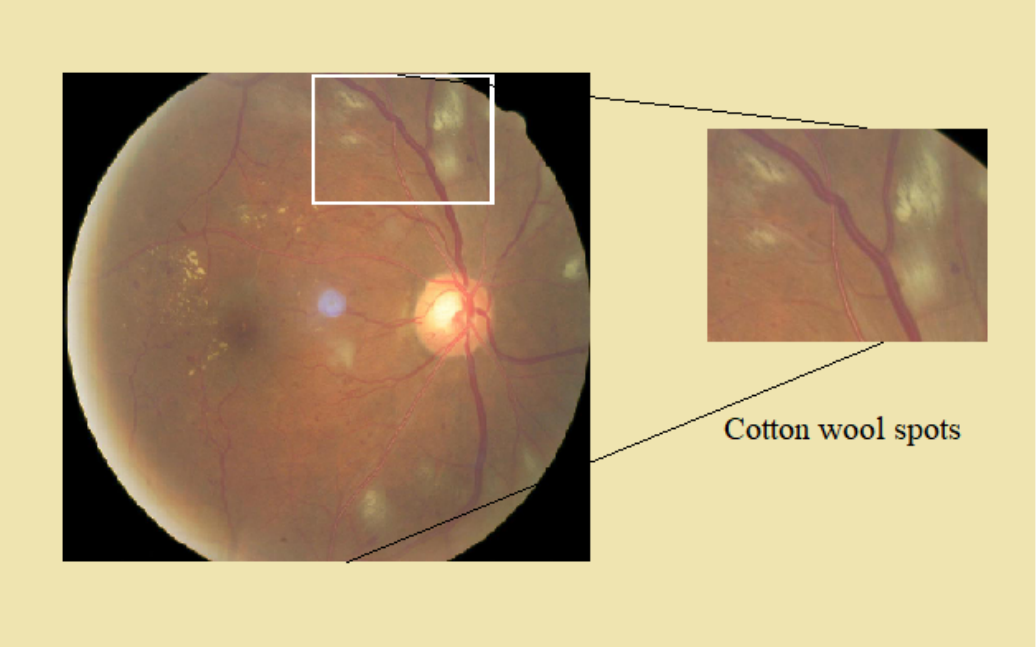


Figura 1.3: Imagen de fondo de ojo con exudades algodonosos.

En la figura 1.4 se muestra un ejemplo de las lesiones denominadas venous beading, que se caracterizan por cambios bruscos en el ancho de las venas, causadas por el debilitamiento de las paredes de las mismas.

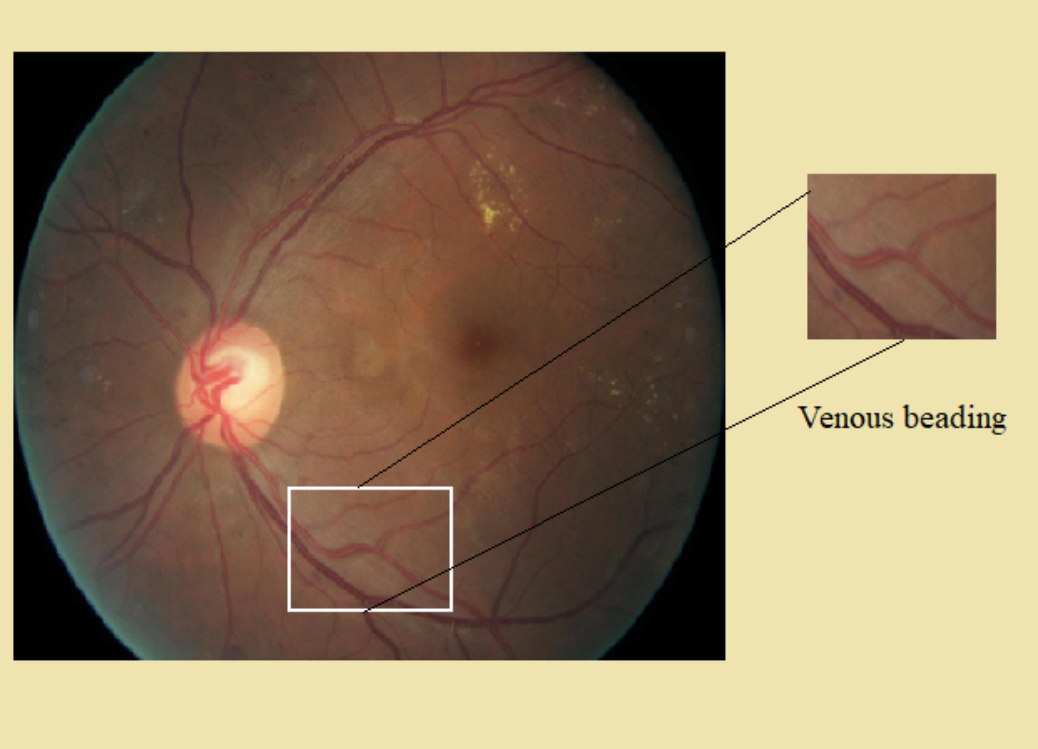


Figura 1.4: Imagen de fondo de ojo con venous beading.

En la figura 1.5 se muestra la un ejemplo de una anomalía microvascular intraretinal o IRMA, que se observan como desviaciones o ramificaciones anormales, siendo dilataciones de capilares para lograr una mejor irrigación sanguínea en la zona. Se las identifica por tener forma de bucles.

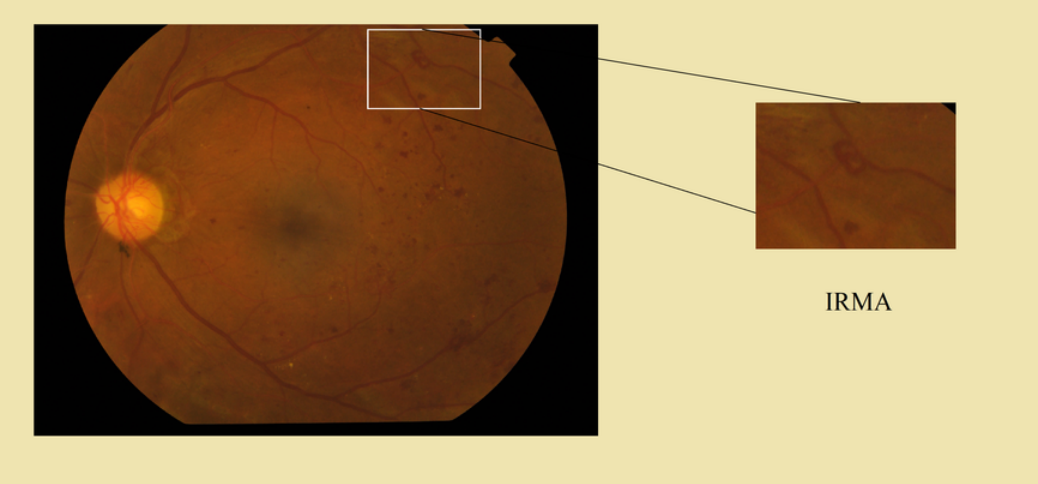


Figura 1.5: Imagen de fondo de ojo con IRMA.

La presencia de IRMA, venous beading o exudados algodonosos clasifican a un paciente en una etapa 3 o una etapa severa de la retinopatía diabética, siendo hasta este momento no proliferativa. Sin embargo, para esta instancia el paciente tiene una cantidad de daño considerable en el tejido retinal. Los signos mostrados a continuación, las hemorragias intraretinales y las neovascularizaciones, se asocian a una etapa de retinopatía proliferativa, comprometiéndose en este estadío la visión de manera irreversible.

En la imagen 1.6 se muestra una hemorragia intraretinal, siendo un signo de un debilitamiento extremo de los vasos sanguíneos. Dependiendo de la gravedad de la hemorragia, los pacientes pueden ser tratados con láser para sellar los capilares comprometidos.

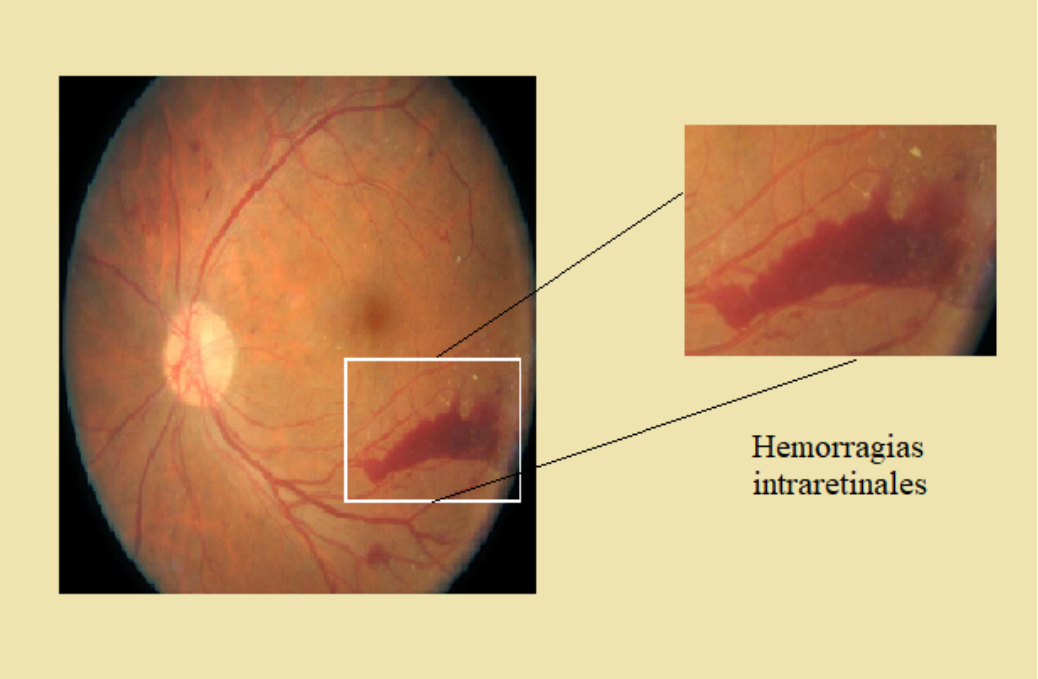


Figura 1.6: Imagen de fondo de ojo con una hemorragia intraretinal.

Finalmente en la figura 1.7 se muestran neovascularizaciones, es decir, la formación de nuevos vasos sanguíneos. Estos son defectuosos y frágiles, recordando su forma a la de corales de mar.

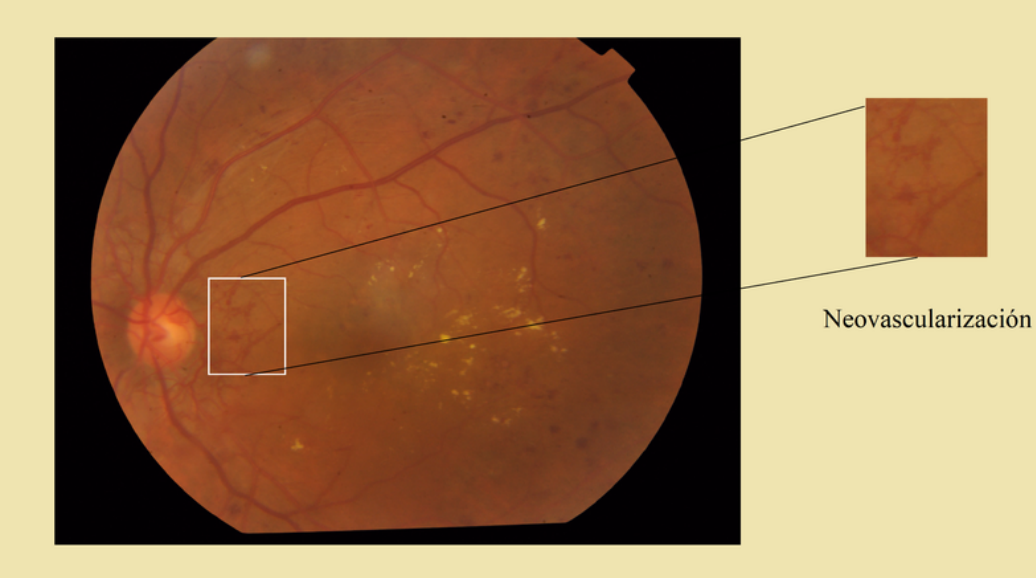


Figura 1.7: Imagen de fondo de ojo con neovascularizaciones.

Capítulo 2

Representación de conocimiento en inteligencia artificial

Con el nacimiento de la inteligencia artificial se lograron resolver rápidamente gran cantidad de problemas de dificultad media para los humanos, pero que sin embargo eran relativamente sencillos de resolver empleando computadoras. La mayoría de estos problemas tenían una característica en común: sus soluciones podían definirse como una lista de tareas secuenciales. El verdadero desafío, sin embargo, apareció con el surgimiento de la necesidad de resolver problemas donde la secuencia de pasos que realiza el humano para resolverlos no es tan directa ni tan simple de detallar. En este sentido, algoritmos de inteligencia artificial que usan redes neuronales en sus modelos se empezaron a utilizar, tratando de encontrar una solución a este tipo de problemas. Particularmente eran de interés modelos que intentaban aprender de la experiencia, es decir, que buscaban patrones en los datos para tratar de encontrar criterios, con el fin de aprender a realizar una determinada tarea de manera satisfactoria.

Las dificultades que sugían en modelos que dependían casi exclusivamente

de lo que se denomina *feature engineering*, es decir, donde las variables relevantes al problema son dadas de antemano como entradas, dieron la pauta de que estos nuevos modelos, donde las tareas de detección de información relevante y la resolución del problema eran resueltas simultáneamente, podrían mejorar los resultados obtenidos hasta la fecha, e incluso, ampliar el espectro de capacidad de resolución de problemas del área de la inteligencia artificial, ya que para muchos problemas, el acceso a las variables relevantes no es tan simple.

Esta área, en la cual los modelos se dice que aprenden de la experiencia, se denomina machine learning. Las soluciones obtenidas con este tipo de modelos dependen fuertemente de las representaciones formadas a partir de los datos dados, por lo que el enfoque de este campo es hacer que el sistema extraiga las variables relevantes, para después utilizarlas en la resolución del problema. Este enfoque se de denomina representational learning.

La gran cantidad de factores que agregan varianza a los datos utilizados en problemas reales, hace que sea fundamental que los modelos que utilicen estos datos para la resolución del alguna tarea reconozcan y separen el ruido y variables que no tienen relevancia en la resolución del problema, de las variables importantes. La anterior tarea de reconocer representaciones útiles en presencia de ruido en general no es simple, como por ejemplo reconocer objetos en imágenes que tienen distinto fondo, iluminación, tamaño, orientación, etc. Resolver tareas de tal magnitud, utilizando un modelo que requiera indicación explicita tanto de las variables relevantes como de las que no aportan a la resolución de problema, puede llegar ser muy complicado y no es claro que sea viable en muchos casos. En este sentido, el campo de Aprendizaje Profundo o Deep Learning propone resolver este inconveniente aprendiendo representaciones de alto nivel a partir de representaciones de más bajo nivel, utilizando el concepto de jerarquía representacional, formando represenciones más complejas a partir de represenciones más simples. A continuación se describen distintos modelos que, en orden creciente de complejidad, han aportado al problema de representación y aprendizaje de funciones cognitivas

complejas.

2.1. Perceptrón

El perceptrón es un modelo de discriminante lineal desarrollado por Rosenblatt [2] que ocupó un lugar importante en la historia de algoritmos de reconocimiento de patrones. Corresponde a un modelo de clasificación binaria en el cual se le aplica una función no lineal $\phi(\cdot)$ al vector de entrada $\mathbf{X}$, utilizando posteriormente esta entrada tranformada para construir la salida de la forma mostrada en la ecuación (2.1).

$$y(\mathbf{x}) = f(\mathbf{w}^T \phi(\mathbf{X})) \tag{2.1}$$

donde $\mathbf{w}$ son parámetros a determinar y la función $f(\cdot)$ es la función escalón, dada por

$$f(a) = \begin{cases} -1, & a < 0 \\ +1, & a \geqslant 0 \end{cases}$$

Normalmente el vector $\phi(\mathbf{X})$ incluye el componente de bias $\phi_0(\mathbf{X}) = 1$. El perceptrón puede ser representado por el diagrama mostrado en la figura 2.1, donde en este ejemplo se tienen 5 entradas y ϕ es la función identidad, siguiendo el caso del modelo original de perceptrón.

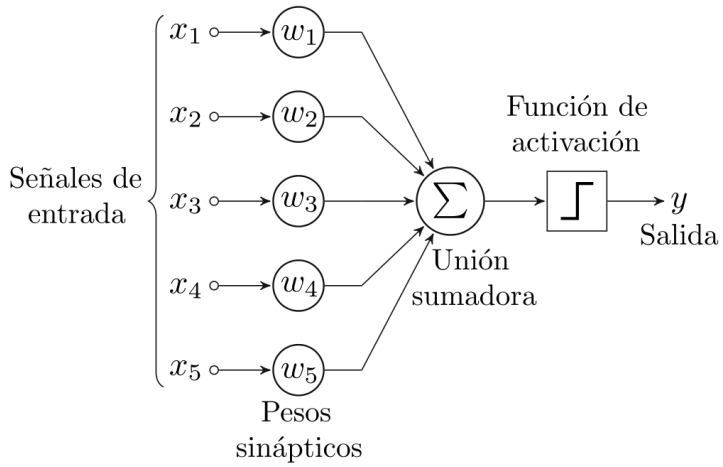


Figura 2.1: Esquema de un perceptrón con 5 entradas.

El objetivo del algoritmo es encontrar el conjunto de pesos que permita que la unidad de salida separe el espacio de entradas posibles en dos soluciones. En caso de que el problema sea linealmente separable, es posible desarrollar un algoritmo que va ajustando paulatinamente el valor de los pesos a partir del aprendizaje supervisado, es decir, emplear un número limitado de ejemplos para los cuales se conoce la respuesta deseada.

Se puede considerar al perceptrón como el precursor de los módelos que usan redes neuronales, donde las ideas de buscar la importancia de las variables de entradas mediante la determinación de los pesos y la aplicación de funciones no lineales juegan un papel central, siendo su finalidad evidente en el modelo del perceptrón, pero no tan evidente en modelos de redes neuronales con muchas capas y funciones de activación más complejas.

2.2. Redes Neuronales

El término *redes neuronales* engloba una serie bastante amplia de modelos y métodos de aprendizaje, estando los modelos empleados en este trabajo dentro de esta categoría. Tomando las ideas del perceptrón, el propósito

de las redes neuronales es la extracción de características de las variables de entrada, modelando la variable de salida como una función no lineal de dichas características, como se muestra en la ecuación (2.2).

$$y(\mathbf{X}, \mathbf{W}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\mathbf{X})\right) \tag{2.2}$$

donde $f(\cdot)$ es una función de activación no lineal en el caso de clasificación. Extendiendo este modelo se llega al modelo básico de red neuronal, haciendo que las funciones $\phi(\mathbf{X})$ dependan de distintos parámetros para luego ajustarlos, junto con los coeficientes w_j, durante el entrenamiento.

El modelo básico de red neuronal empieza por lo tanto con la construcción de M combinaciones lineales de las variables de entrada x_1, x_2,..., x_D de la forma mostrada en la ecuación (2.3).

$$a_j = \sum_{i=1}^{D} w_{ij}^{(1)} x_i + w_{j0}^{(1)} \tag{2.3}$$

con $j = 1, 2, ..., M$ y el supraíndice (1) indica que se trata de parámetros correspondientes a la primera capa de la red. Los parámetros $w_{ji}^{(1)}$ se denominan pesos, mientras que al parámetro $w_{j0}^{(1)}$ se lo denomina bias. A las cantidades a_j se las conoce como activaciones y se obtiene la salida de la capa al aplicarles una función de activación no lineal $h(\cdot)$, como se muestra en la ecuación (2.4).

$$z_j = h(a_j) \tag{2.4}$$

Inicialmente, las funciones no lineales $h(\cdot)$ eran generalmente funciones sigmoidales, sin embargo, con el paso del tiempo estas funciones utilizadas en redes neuronales artificiales fueron evolucionando, buscando propiedades particulares, ya sea para características en el modelo o para mejoras en el entrenamiento. Actualmente una de las más utilizadas es la función de unidad lineal rectificada o ReLU, o variantes de la misma. Esta función es igual a la función identidad para valores mayores que cero, por lo que a diferencia de las funciones sigmoidales, su gradiente no se hace nulo para valores de

entrada grandes.

Suponiendo por cuestiones de simplicidad que nuestra red tiene 2 capas, una vez obtenidos los valores de salida z_j, nuevamente se combina a estos linealmente para obtener los valores denominados unidades de activación de salida a_k, como se muestra en la ecuación (2.5).

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{2.5}$$

donde $k = 1, 2,..., K$, y k es el número total de outputs. Finalmente, las unidades de activación de salida son transformada utilizando una función de activación apropiada para obtener las salidas de la red y_k. La elección de la función de activación de la salida de la red depende de la naturaleza de los datos y de la distribución propuesta o asumida para las variables targets.

Combinando las etapas anteriores, obtenemos la función de salida de toda la red, a partir de los parámetros de entrada, mostrada en el ecuación (2.6).

$$y_k(\mathbf{X}, \mathbf{W}) = \sigma \left(\sum_{j=1}^{M} w_{kj}^{(2)} h \left(\sum_{j=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right) \tag{2.6}$$

Se puede observar que la red neuronal es simplemente una función no lineal de una serie de variables de entrada x_i, obteniendo la variable de salida y_i, controlada por un set de parámetros ajustables $\mathbf{W}$. Esta función puede ser representada en forma de diagrama como se muestra en la figura 2.2.

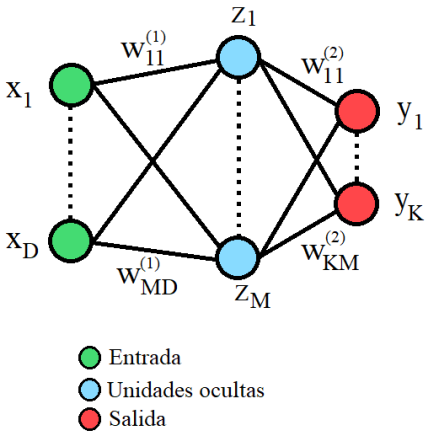


Figura 2.2: Diagrama de una red neuronal de 2 capas

El proceso de evaluación mostrado por la ecuación 2.6 se puede considerar como una propagación hacia adelante de la información a través de la red. Se observa que para el ejemplo dado, dicha propagación se compone de 2 etapas, cada una de las cuales se parece a la propagación de información en un perceptrón, por lo que esta red se denomina perceptrón multicapa o MLP por sus siglas en inglés. Esta red, sin embargo, usa funciones de activación no lineales continuas, mientras que en el perceptrón la función de salida final es una función discontinua, la función escalón. Esto último le da una gran ventaja a la red respecto al percetrón, y es la posibilidad de calcular la derivada parcial de las salidas respecto de cada uno de los pesos w_i, siendo esto importante a la hora de entrenar la red y ajustarlos.

Normalmente, las redes neuronales pueden tener desde unas pocas capas hasta el orden de centenas, con distinta cantidad de unidades cada capa. La elección de la cantidad de capas y de la cantidad de unidades por capa no es algo sencillo de determinar, siendo siempre conveniente empezar con modelos simples, de pocas unidades y pocas capas, e ir complejizando la arquitectura.

Parámetros antes mencionados como la cantidad de capas y el número

de unidades por capas, cuyos valores deben ser elegidos antes de empezar propiamente con el entrenamiento para el ajuste de los pesos, y que además forman parte de la configuración del modelo, se denominan hiperparámetros. La elección de estos parámetros es una etapa fundamental, ya que ellos determinan gran parte de las características del modelo, del aprendizaje y la capacidad del mismo, teniendo un impacto directo en la precisión máxima teórica del modelo y el camino de aprendizaje seguido durante el entrenamiento. Los valores óptimos de los hiperparámetros dependen del problema en cuestión y la elección de un hiperparámetro posiblemente condicione el valor óptimo de otro hiperparámetro, por lo que su determinación no es una tarea secilla. El área de optimización de hiperparámetros es extensa en sí misma y en constante actualización. Como valores iniciales generalmente se elijen valores de hiperparámentros de otros problemas similares, así como arquitecturas que tuvieron éxito en el pasado, pudiendo ser optimizados posteriormente utilizando distintas técnicas encontradas en la literatura.

2.3. Entrenamiento de redes neuronales y aprendizaje de los pesos

Las redes neuronales tienen parámetros, los pesos, que en principio son desconocidos. La finalidad del entrenamiento de la red es ajustar los pesos de manera de lograr resolver de manera efectiva una determinada tarea.

Para monitorear el desempeño del modelo durante el entrenamiento se elige una función error, la cual nos indica qué tan bien (o mal) realiza la tarea el modelo. La elección de la función depende del problema en cuestión, y en problemas de regresión o clasificación se elige una función que tenga en cuenta la distancia entre los valores que tiene como salida nuestro modelo y los valores que debería tener. Para el caso de regresión, la suma de los errores al cuadrado es una medida del error en el ajuste, mientras que para nuestro caso, la tarea de clasificación, se usa la entropía cruzada, mostradas ambas

funciones en las ecuaciones 2.7 y 2.8 respectivamente.

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2 \tag{2.7}$$

$$R(\theta) = -\sum_{k=1}^{K} \sum_{i=1}^{N} y_{ik} \log f_k(x_i) \tag{2.8}$$

El enfoque genérico para minimizar $R(\theta)$ es utilizar gradientes descendentes, empleando en redes neuronales la técnica de back-propagation [3] o retropropagación de errores para hallar que impacto tiene en la salida un determinado peso de la red, haciendo uso de la propiedad de la red de ser una composición de funciones. La tarea de actualizar los pesos es función del optimizador, habiendo muchas variantes. Uno de los más simples es el llamado Gradientes Descendientes Estocástico o SGD por sus siglas en inglés, en el cual se toman batchs aleatorios del set de entrenamiento y se actualizan los pesos utilizando una fracción del gradiente, llamando a la contante que determina esta fracción tasa de aprendizaje. Una variante de esto es, además de utilizar información del gradiente actual, restarle un valor proporcional a la última actualización de los pesos. La constante de proporcionalidad utilizada para este fin se denomina momento, por su analogía con el significado de este término en el campo de la física. El optimizador descrito anteriormente es el empleado en este trabajo.

Una vez que los pesos fueron actualizados pasando por todas las imágenes del dataset de entrenamiento, se dice que ha pasado una época del entrenamiento, siendo esta una medida común de este proceso.

El desafío que enfrenta cualquier modelo en machine learning es tener un buen desempeño en datos que nunca vio, esto es, tener una buena generalización. Para ello, se utilizan distintas técnicas para evitar el sobreajuste, presentando a continuación las empleadas en el presente trabajo. En general, este tipo de técnicas son restricciones o penalidades a los valores de los pesos, prefiriendo de esta manera los modelos más simples.

2.4. Técnicas de regularización

2.4.1. Regularización L1 y L2

Una de las técnicas más conocidas de regularización es la técnica de *weight decay* o regularización de pesos. En ella, se modifica la función de error o función de costo a minimizar, agregando un término que tiene en cuenta el valor de los pesos, por lo que al entrenar son preferidos los módelos que tienen pesos de menor valor. En la ecuación 2.9 se muestra particularmente la función de error al utilizar MSE y regularización L2, donde se le agrega a la función error tradicional un término proporcional a la suma del cuadrado de los pesos.

$$J(\mathbf{w}) = MSE_{train} + \lambda \mathbf{w}^T \mathbf{w} \tag{2.9}$$

En la ecuación 2.9 la constante λ es llamada constante de regularización y en general es un hiperparámetro importante a la hora de entrenar el modelo. Esta técnica de regularización, al agregar un término a la función error, puede ser pensada como una restricción más en los valores que pueden tomar los parámetros a ajustar.

En nuestro caso, al aplicar transferencia de aprendizaje y estar trabajando en el rango de un set de datos de muy pocas imágenes, este tipo de regularización es fundamental para una buena generalización al realizar el entrenamiento, ya que al tener pocas imágenes, la red tiende a aprender de memoria los datos de entrenamiento y tener una precisión baja para los datos de validación.

2.4.2. Dropout

Esta técnica consiste en anular temporalmente la salida de algunas unidades de la red. La idea es asignarle cero, en cada época del entrenamiento, a distintas unidades elegidas de manera aleatoria, entrenando por lo tanto redes más angostas, es decir, con menos cantidad de unidades en las capas con dropout. La motivación para la utilización de esta técnica [4] es promover el aprendizaje de características que sean útiles en la tarea a resolver, pero que además la red aprenda a resolver la tarea utilizando una selección aleatoria de pesos, evitando de esta manera la dependencia de conexiones específicas para un buen desempeño.

2.4.3. Data Augmentation

Idealmente, cuando entrenamos los modelos de redes neuronales, suponemos que el set de datos utilizado en el entrenamiento es una muestra representiva y que tiene suficiente información de manera se puede inferir a partir de ellos la variable de salida. Muchas veces, la cantidad de datos disponible es limitada, como lo es en nuestro caso. Sin embargo, es posible aplicar transformaciones a los datos a disposición y generar datos que, si bien son ficticios, son válidos como datos de entrada y además es completamente factible que dicho dato sea producto de la función generadora de datos, la cual generalmente se desconoce.

En muchos problemas, las transformaciones a aplicar a los datos de entrada para generar datos verosímiles, que a su vez están asociadas a simetrías en los dichos datos, no son sencillas de encontrar. Sin embargo, cuando los datos de entrada son imágenes, algunas transformaciones pueden llegar a ser evidentes, como lo son por ejemplo pequeñas rotaciones, pequeños cambios en el brillo, pequeños cambios de contraste e incluso el agregar un poco de ruido a las imágenes, pero esto depende mucho del problema en cuestión, ya que transformaciones que pueden ser útiles ayudando a la generalización en

un problema, pueden ser perjudiciales para otro.

En nuestro caso, la generalización de las redes entrenadas se ven beneficiadas por el uso de esta técnica, particularmente espejando, tanto horizontal como verticalmente las imágenes de fondo de ojo, haciendo pequeñas rotaciones, haciendo pequeños cambios en el brillo y aplicando zoom a las imágenes.

2.5. Redes Convolucionales

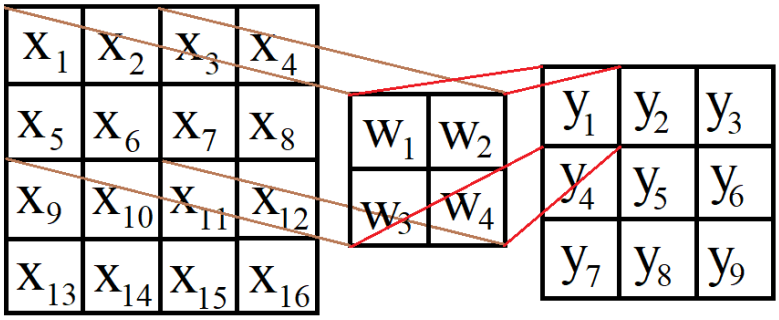
Las redes convolucionales tienen gran importancia en el campo de deep learning, ya que estas fueron los primeros modelos utilizados que obtuvieron excelentes resultados en el campo de reconocimiento de imágenes, como por ejemplo en los años 90 en el grupo AT&T para reconocimiento de cheques y en los años 2000 en el grupo Microsoft para reconocimiento de dígitos manuscritos.

En el campo de computer vision, las redes convolucionales han tenido gran relevancia en los últimos años, logrando precisiones que superan a la humana en tareas de clasificación y reconocimiento de imágenes. El punto de quiebre se considera que fue en el año 2012, cuando la red AlexNet obtuvo un error top-5 de 15,3 %, más de 10 % menos que el competidor que menor error tenía hasta la fecha, donde error top-5 es la fracción de los resultados para los cuales la categoría correcta no se encuentra entre las 5 categorías más probables predichas por el modelo.

A la hora de reconocer imágenes, varias propiedades fueron tenidas en cuenta para proponer el modelo de red convolucional y, una forma de crear modelos que cumplan con ciertas propiedades o que sean invariantes ante ciertas transformaciones, es hacer que dichas invarianzas sean una propiedad de su estructura, siendo este el caso de las redes convolucionales. Al proponer este tipo de modelos para reconocer imágenes, se tuvo en cuenta que estos tenían que ser invariantes ante translaciones, escaleos y pequeñas rotaciones.

Además, otra propiedad fundamental que fue tenida en cuenta es el aumento de la correlación entre los pixeles con su cercanía. A causa de esto último, se intentan buscar entonces características que sean locales, tomando pequeñas regiones de la imágenes y extrayendo propiedades de ella, combinándolas posteriormente en capas más profundas de la red para tener información de la imagen como un todo. Para lograr todo lo anterior, las redes convolucionales hacen uso de campos receptivos locales, compartiendo pesos al aplicar filtros y finalmente subsampleando.

Generalmente los bloques o grupo de capas de las redes convolucionales que se encargan de la extracción de las características se componen de una o varias capas convolucionales, seguidas por capas de subsampleo, reduciendo de esta manera la dimensionalidad en ancho y alto de las imágenes a medida que avanzan en la red. En las figuras 2.3 y 2.4 se muestran un esquema de una convolución y un ejemplo de una capa de subsampleo o pooling, con un kernel de 2x2, tomando en este ejemplo el máximo de cada subregión de 2x2 pixeles. Otras opciones de subsampleo toman el promedio de la subregiones en lugar del máximo, todos con la misma finalidad de reducción de la dimensionalidad. A la forma de subsampleo mostrada en la figura 2.4 se la llama comúnmente max-pooling.



$$y_1 = x_1 w_1 + x_2 w_2 + x_5 w_3 + x_6 w_4$$

Figura 2.3: Esquema ilustrativo de una convolución.

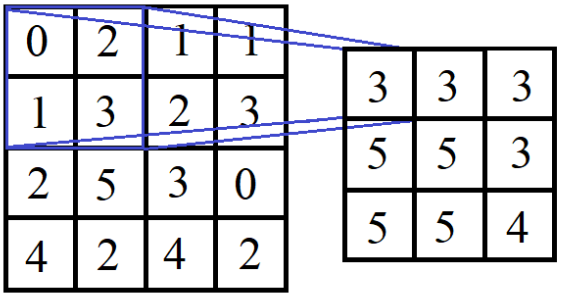


Figura 2.4: Ejemplo de la aplización de una capa de max-pooling.

En la capa convolucional las unidades están organizadas en planos, es decir, que todas las entradas que llegaron hasta esa instancia tienen la misma cantidad de transformaciones aplicadas. A la salida de cada uno de estos planos se la denomina mapa de características. Cada unidad en los mapas de caraterísticas toman solo subregiones de la entrada, teniendo además la propiedad de compartir los pesos todas las unidades de ese mapa, por lo que se pueden considerar como detectoras de una misma característica en

cada subregión, recorriendo toda la imagen en su búsqueda. A su vez, la evaluación de cada unidad puede ser vista como una convolución, donde a cada subregión se le aplica un filtro, también llamado kernel, y de allí su nombre.

Típicamente, el paso por un bloque convolucional consiste generalmente de 3 etapas: en la primera se realizan las convoluciones en paralelo para producir activaciones lineales, posteriormente a estas se les aplica una función de activación no lineal, para finalmente pasar por una etapa de subsampleo o pooling. El subsampleo da la propiedad de invarianza ante pequeñas transalaciones, ya que en estas se toman como entrada las activaciones de los mapas de características, donde cada una de las unidades de esta capa, al igual que en la capa convolucional, recibe como entrada una pequeña región de la capa anterior. Existen distintas formas de realizar este subsampleo y uno de los más utilizado es el maxpooling.

En una red convolucional típica hay varios bloques de capas convolucionales y de subsampleo, con las últimas capas de la red de tipo densas, es decir, que todas las unidades de la capa anterior se conectan cada unidad de dicha capa. Las redes preentrenadas en nuestro caso tienen del orden del centenar de capas pertenecientes a bloques convolucionales, con las últimas dos capas de tipo densas. En la última capa, la capa de salida, la función de activación utilizada para el caso de clasificación es del tipo softmax o exponencial normalizada, como se muestra en la ecuación (2.10), utilizada como salida para variables de tipo categórica.

$$\sigma = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{2.10}$$

Existen muchos tipos distintos de arquitecturas, con distintos tipos de sequencialidad de capas, distinto número de parámetros y distintos tipos de conexiones entre capas. A continuación se presentan las redes preentrenadas utilizadas en el presente trabajo y algunas de sus principales caraterísticas.

2.5.1. DenseNet

Redes convolucionales densamente conectadas o DenseNets fueron introducidas recientemente [5] como solución al problema de la disminución del valor del gradiente al entrenar usando gradientes descendentes. En la publicación original proponen como una posible solución una arquitectura de red de tipo DenseNet, asegurando que de esta manera el flujo de información es máximo entre las capas de la red, conectando todas las capas directamente entre ellas, matcheando el tamaño de los mapas de características. A diferencia de otro tipo de redes, las residual networks, en las que los features se suman entre sí, en las redes de tipo DenseNet se concatenan. Aseguran además que esta red requiere menos cantidad de parámetros que las redes convolucionales tradicionales, ya que no es necesario reaprender mapas de características de forma redundante. En la figura 2.5 se muestra un esquema de un bloque denso de 5 capas, observando que todas las capas se conectan con las anteriores en el bloque.

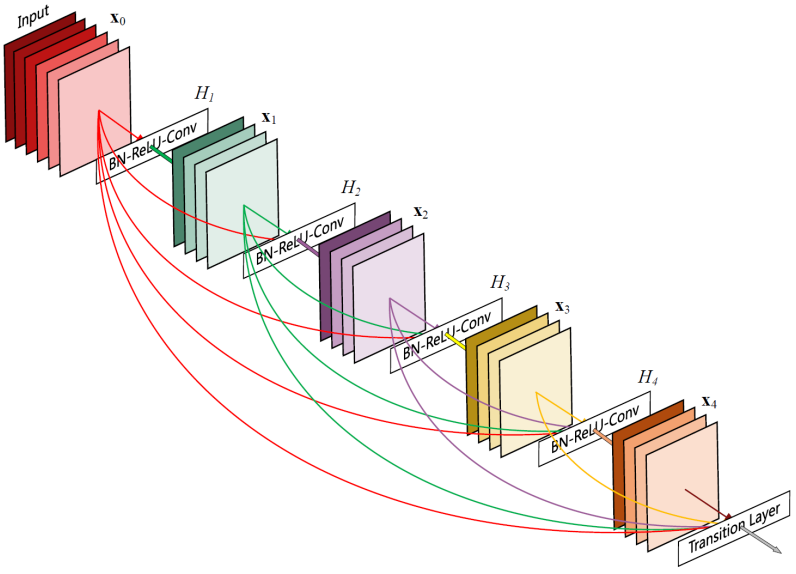


Figura 2.5: Diagrama de un bloque denso con 5 capas y una tasa de crecimiento de k=4. (Imágen obtenida de [5])

En la figura 2.6 se muestra esquemáticamente la arquitectura de la red

DenseNet, observando que se compone de muchos bloques convolucionales densos, donde AVGPool hace referencia de una capa de pooling en la cual se promedia cada subregión de la salida de la capa anterior. Se observa la red termina con una capa tipo densa y con una activación de tipo softmax.

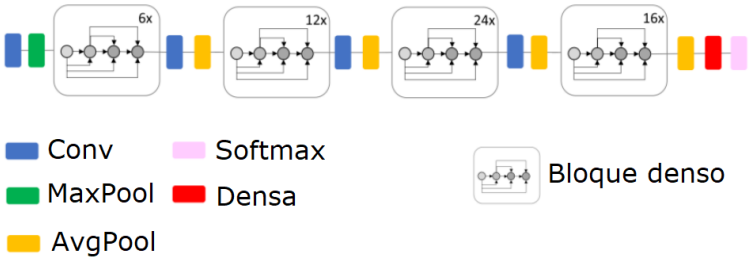


Figura 2.6: Esquema de la arquitectura de la red DenseNet.

2.5.2. Inception V3

Una forma de intentar mejorar la precisión de una red puede ser aumentar su profundidad y el número de unidades por capa, es decir aumentar la capacidad de la red. Sin embargo, la cantidad de parámetros es un gran cuello de botella en el entrenamiento, además, al hacer esto siempre se tiene el compromiso de sobreajuste, es decir, que la red tenga más capacidad o que pueda guardar más información que la información relevante extraible del dataset utilizado. La forma de afrontar el problema propuesto por el modelo de red de tipo Inception [6], es tener en su estructura conexiones dispersas y poco densas. La primera versión de esta red fue llamada GoogLeNet, inspirando con su arquitectura muchas otras redes posteriormente y teniendo suma relevancia en el mundo de reconocimiento de imágenes.

Las redes tipo Inception se mejoraron con el tiempo, primero con el agregado de la técnica de batch normalization [7], en la cual se normaliza, en caso de ser necesario, la entrada de cada bloque convolucional de manera de evitar la saturación de las funciones de activación sigmoidales y por lo tanto, evitar

tener gradientes de cada vez menor valor, acelerando de esta forma el proceso de aprendizaje de la red. A esta versión de la red se la llamó Inception V2. Posteriormente se agregaron a la arquitectura de la red las ideas de factorización, llamando a esta versión Inception V3, una de las redes preentrenadas utilizadas en el presente trabajo. En la figura 2.7 se muestra un esquema de la arquitectura de la red Inception V3.

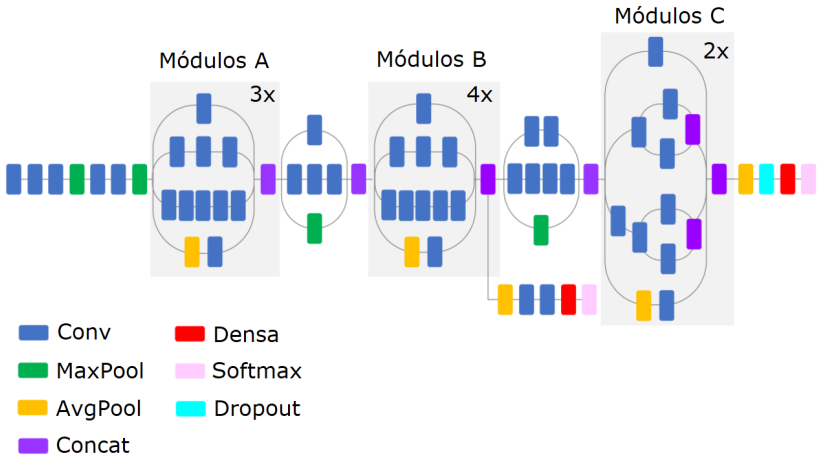


Figura 2.7: Esquema de la arquitectura de la red Inception V3.

La idea de factorizar convoluciones consiste en reducir la cantidad de paramétros y conexiones empleadas en la red sin afectar la eficiencia de la misma. La primera técnica utilizada es el reemplazo de convoluciones con kernels altos, es decir, que tomaban una subregión con muchos pixeles, por sucesivas convoluciones con kernes más bajos, tomando por lo tanto subregiones de menor tamaño. En la figura 2.8 se muestra un ejemplo de un reemplazo de una convolución de 5x5 por dos convoluciones de 3x3 sucesivas.

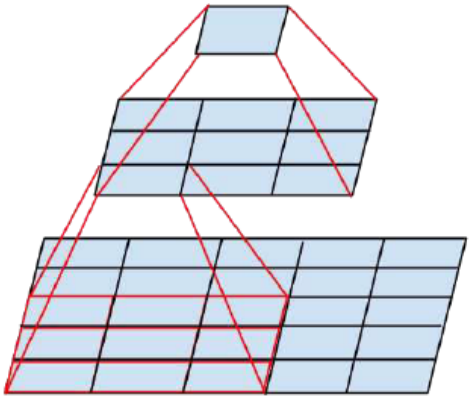


Figura 2.8: Reemplazo de una convolución de 5x5 por dos convoluciones de 3x3.

Vale la pena notar que usando una sola convolución de 5x5 la cantidad de parámetros es 25, mientras que usando dos convoluciones de 3x3 la cantidad de parámetros a aprender son 18, reduciendo la cantidad de parámetros en un 28 %. Esta reducción de parámetros se emplea en los *Módulos A* de la red mostrados en la figura 2.7, mostrando su arquitectura en la figura 2.9. Se indica además en la figura que en la primera versión de la red, GoogLeNet, el kernel de la convolución era efectivamente de 5x5.

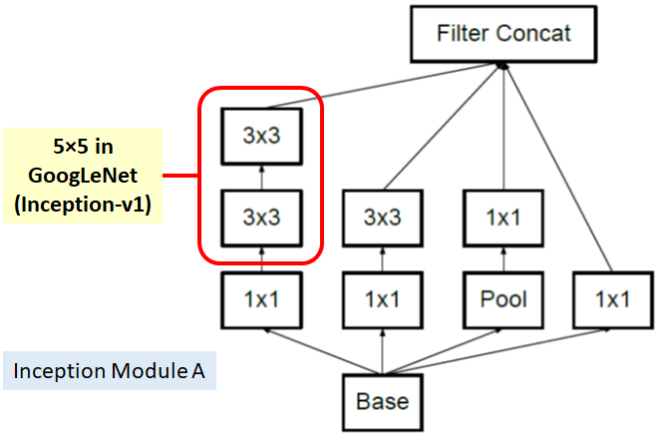


Figura 2.9: Arquitectura del módulo A de la red Inception V3.

Otro tipo de factorización empleada en esta red, es el reemplazo de convoluciones con kernels cuadrados por convoluciones con kernels asimétricos, también logrando de esta manera la reducción de parámetros a ajustar. En la figura 2.10 se muestra como dos convoluciones con kernels asimétricos, particularmente uno de 3x1 seguido por uno de 1x3, reemplazan una convolución con un kernel de 3x3.

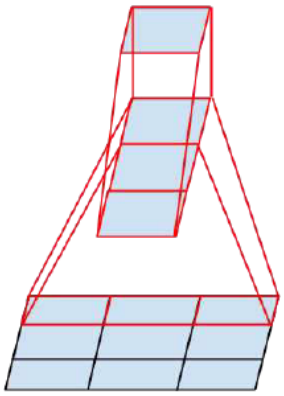


Figura 2.10: Ejemplo de convolución asimétrica.

En el ejemplo de la figura 2.10 se reemplaza una convolución de 3x3 que requiere aprender 9 pesos, por dos convoluciones que requieren en total 6, con una reducción de 33 % de los pesos a entrenar. Este tipo de factorización se encuentra en los *Módulos B* de la red Inception V3, mostrando la arquitectura del mismo en la figura 2.11, donde particularmente se reemplazan convoluciones 7x7 por sucesivas convoluciones asimétricas.

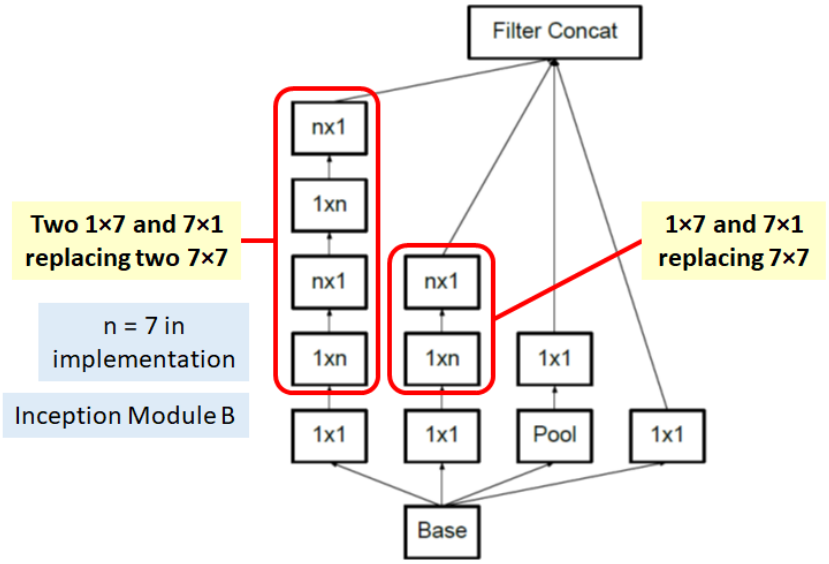


Figura 2.11: Arquitectura de los módulos tipo A de la red Inception V3.

Finalmente, el último módulo agregado es el *Módulo C*, el cual se muestra en la figura 2.12, observando que se agregan convoluciones en paralelo, afirmando que esto beneficiaría la promoción de múltiples representaciones de alto nivel.

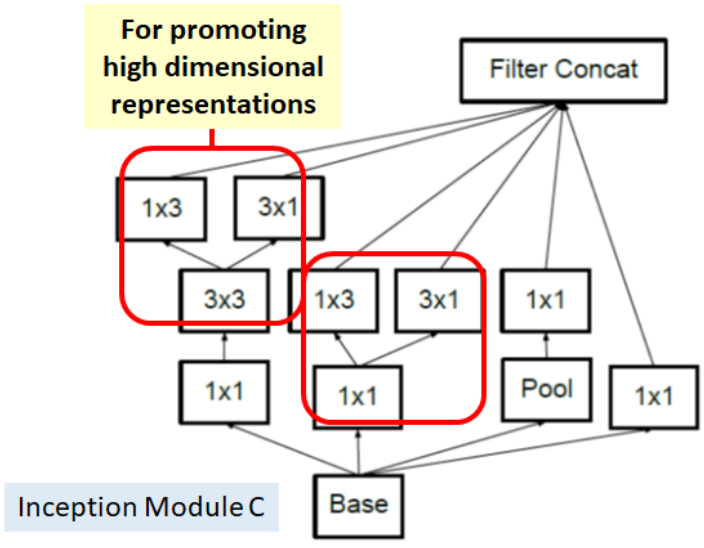


Figura 2.12: Arquitectura de los módulos tipo C de la red Inception V3.

La hipótesis detrás de los módulos de la red Inception es que las correlaciones espaciales y entre canales están suficientemente desacopladas, por lo que es preferible no mapearlas en conjunto. Esto lo logran realizando primero convoluciones 1x1, tomando regiones de un pixel, buscando correlaciones entre canales y posteriormente buscando las correlaciones espaciales con convoluciones de 3x3, 5x5 o 7x7, como es el caso de *Módulo A* mostrado en la figura 2.9.

Esta última hipótesis se hace más fuerte en la red que se muestra a continuación, en la cual se hace la suposición de que correlaciones y espaciales están completamente desacolpadas. Esta red recibe el nombre de Xception, por *Extreme Inception*.

2.5.3. Xception

En la red presentada anteriormente, la red Inception, se tenía como hipótesis el desacople parcial de las correlaciones espaciales y entre canales. Ello equivale a la arquitectura mostrada en la figura 2.13, buscando primero correlaciones entre canales y después correlaciones espaciales.

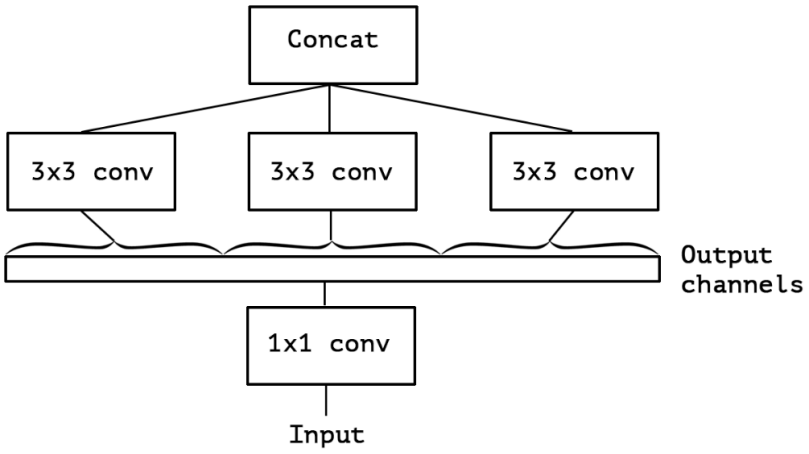


Figura 2.13: Ejemplo esquemático de la hipótesis de desacople en la red Inception. (Imagen obtenida de [6])

Puede hacerse, sin embargo, una hipótesis más fuerte y suponer que las correlaciones espaciales y entre canales pueden mapearse completamente por separado. En la figura 2.14 se muestra un esquema de la arquitectura que supone un desacople completo, donde los filtros que salen de la convolución 1x1 se encuentran esquematizados adyacentemente en una misma capa, y posteriormente se les aplican convoluciones de 3x3 a cada uno.

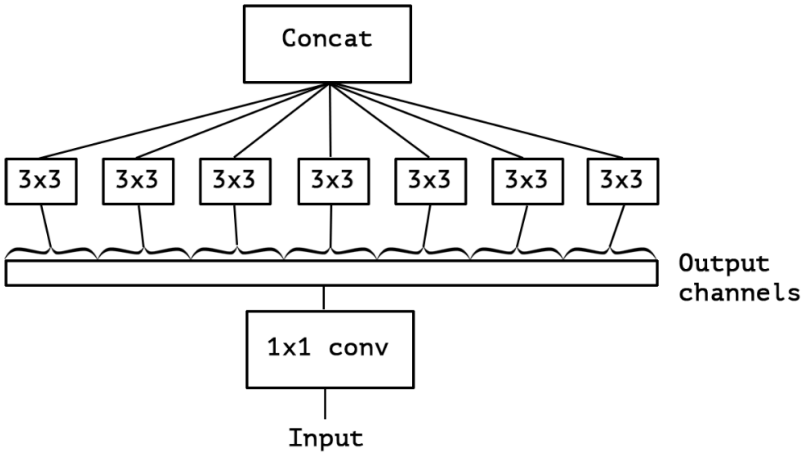


Figura 2.14: Ejemplo esquemático de la hipótesis de desacople completo espacial y entre canales. (Imagen obtenida de [8])

El tipo de convolución antes mostrada se denomina convolución separable en profundidad o *depthwise separable convolution* y es la base de la arquitectura de la red Xception [8], la cual tiene la arquitectura mostrada en la figura 2.15.

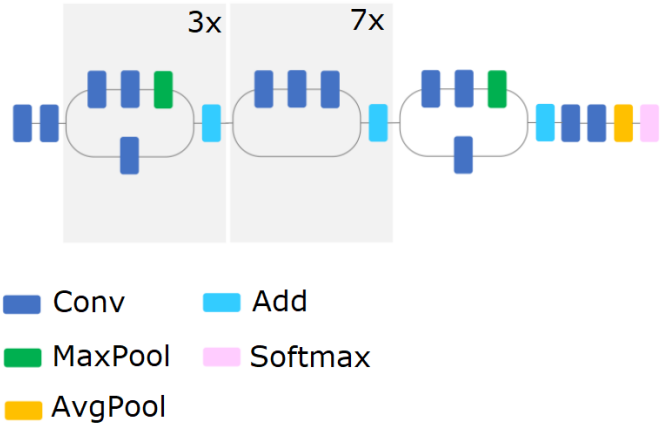


Figura 2.15: Esquema de la arquitectura de la red Xception.

Esta red tiene una cantidad de pesos ligeramente menor que la red Inception V3, con una mayor precisión obtenida en la base de datos ImageNet, por lo que el uso de pesos se realiza de forma más eficiente.

Habiendo visto las características de las redes preentrenadas utilizadas, en la siguiente sección se abordan métodos para el análisis de las redes luego del entrenamiento, particularmente métodos útiles para redes convolucionales y que serán usados posteriormente.

2.6. Análisis las soluciones de las redes entrenadas

Teniendo en cuenta que cada vez se le da más terreno a la inteligencia artificial, con tareas cada vez más dificiles y en áreas donde la confiabilidad se vuelve fundamental, como lo es el área de la medicina, una cuestión que continúa siendo de gran importancia a la hora de utilizar dichos algoritmos es garantizar la confiabilidad y transparencia del mismo. Esto se relaciona

con la interpretabilidad de los pasos seguidos por el algoritmo para resolver la tarea en cuestión, y evaluar el desempeño del mismo con otros parámetros además de la precisión obtenida.

2.6.1. Similaridad en las representaciones

Suponiendo que se entrenan distintas redes para realizar una determinada tarea, algunas preguntas que naturalmente surgen son: ¿Pueden existir varias solucionas al problema o las soluciones encontradas son equivalente? si no es así, ¿las soluciones obtenidas qué tan similares son? y sobretodo ¿qué se puede entender por similaridad en este caso?. En busca de respuestas a preguntas como las planteadas anteriormente, una línea de investigación activa es obtener información sobre la similaridad de las redes entrenadas comparando las representaciones aprendidas por las mismas, ya que a través de ellas la red es capaz de resolver la tarea indicada. Hay que tener en cuenta que es complicado intentar buscar representaciones de conceptos particulares, sobre todo porque estas representaciones son herramientas que la red emplea para resolver el problema, y no es claro que tengan un equivalente a conceptos que utilizamos los humanos, por ejemplo, con el fin de caracterizar objetos.

En vista de que estamos en el campo de reconocimiento de imágenes, es interesante explorar métodos que, visualmente, puedan darmos alguna idea de qué regiones de una imagen la hacen pertenecer a una categoría y no a otra. En nuestro caso, y siguiendo con la búsqueda de mayor confiabilidad, se explora la detección de las lesiones en la retina. Esto brinda información muy valiosa, ya que de esta manera se puede confirmar que la red encuentra correctamente los patrones que comparten las imágenes de una misma clase, siendo en nuestro caso, por ejemplo, las lesiones típicas de la retinopatía diabética. Por otro lado, también se podría a partir de ello, intuir como mejorar la precisión de nuestro modelo. En la siguiente sección se presentan las técnicas utilizadas en este sentido y se introduce conceptualmente a su funcionamiento.

2.6.2. Mapas de activación de clases

Distintos métodos se han empleado para tener un mejor entendimiento del aprendizaje de las redes y, particularmente trabajando con imágenes, cualquier información visual que podamos obtener de la red es interesante de adquirir. Una de las primeras técnicas que permitió obtener información visual clara fue el método de Mapas de Activación de Clases (CAM). El método CAM [9] requiere que la red tenga una arquitectura particular, específicamente que las últimas capas sean un conjunto de filtros convolucionales, seguidos de una capa del tipo Global Average Pooling, en la cual se promedia toda la región del mapa de activación. Luego, es necesario que esta capa sea la entrada de la última capa, la cual se compone de las unidades para clasificación en las distintas categorías.

Un esquema de la técnica se muestra en la figura 2.16, observando que a partir de la suma pesada de los mapas de activación, se obtienen las áreas de la imágen que tienen más importancia a la hora de determinar la clase a la que pertenece una imagen.

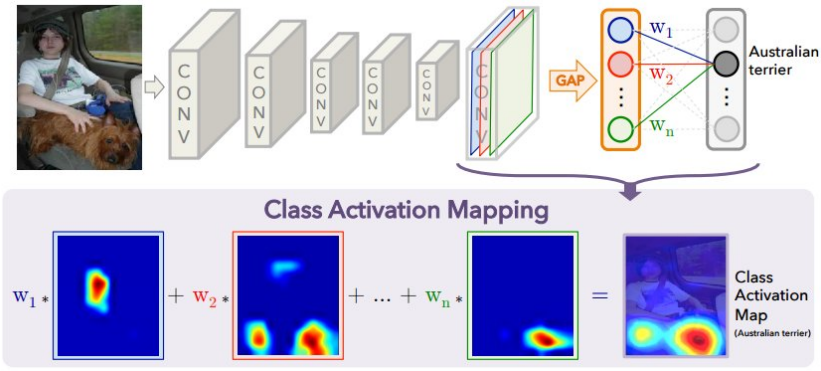


Figura 2.16: Esquema del funcionamiento de la herramienta Mapas de Activación de Clases (CAM). Imagen obtenida de [9].

Uno de los principales inconvenientes del método de Mapas de Activación de Clases es la falta de resolución a la hora de determinar las regiones importantes para una clase dada. Esta inconveniencia se presenta ya que ge-

neralmente la salida de los últimos filtros convolucionales en redes profundas son del orden de la decena de pixeles. El otro gran inconveniente es la arquitectura de red requerida para su utilización, perdiendo de esta manera generalización en la utilización del método.

Siguiendo con esta búsqueda de generalización y teniendo en mente obtener las regiones de la imagen importantes para cada clase, el método de Mapas de Activación de Clases por Gradientes pesados o Grad CAM [10] fue desarrollado para este fin. En este método se aplica el gradiente a la salida de una determinada clase y^c con respectos a mapas de caraterísticas seleccionados. El resultado de esta operación es una medida de la importancia de estos últimos para la clase en cuestión. Con este método por lo tanto gana generalización en su utilización, ya que es independiente de la arquitectura de la red, sin embargo todavía se tiene el problema de la falta de resolución. Para sortear este inconveniente, se realiza una retropropagación guiada. Con lo anterior se obtienen los pixeles que activaron un determinado mapa de activación, y teniendo de antemano la importancia de dicho mapa para una determinada clase con el método Grad CAM, se puede obtener las regiones de importancia con mucha más resolución. En la figura 2.17 se muestra un esquema del método.

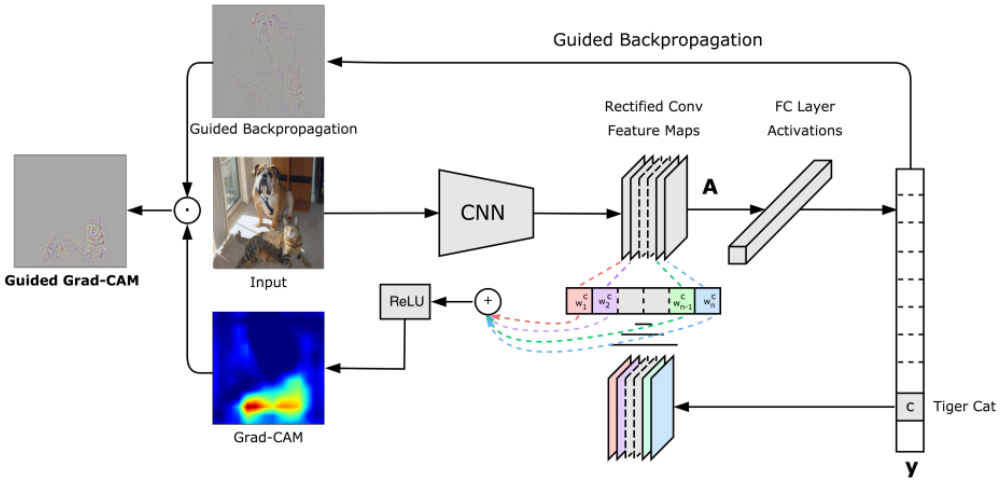


Figura 2.17: Esquema del funcionamiento de la técnica Grad CAM. Imagen obtenida de [10].

2.7. Estado del arte

Actualmente la detección de retinopatía diabética en Argentina se realiza por un especialista en retina, sin ningún tipo de ayuda de sistemas o algoritmos especializados en la detección de lesiones. En el resto del mundo, exiten distintos sistemas que realizan detección automática de retinopatía diabética, y estos se pueden clasificar en dos grandes grupos: los que se basan fuertemente en *feature engineering*, dependiendo fuertemente de especialistas en el campo, para preparar y adaptar funciones específicas para la detección de lesiones particulares. El otro grupo corresponde a algoritmos que usan redes neuronales para la clasificación de las imágenes. En esta sección se describe brevemente el desarrollo de este segundo grupo de algoritmos, con algunos de sus resultados. Existen también algoritmos que combinan la detección de lesiones con datos del paciente para realizar un diagnóstico, como la edad, el tiempo desde que le detectaron diabetes, etc. Sin embargo, estos algoritmos son los menos utilizados.

Uno de los primeros trabajos con resultados prometedores fue el de Abràmoff et al. [11], en el cual se utilizan distintas redes, entre ellas la red AlexNet, para la detección de lesiones particulares en las imágenes de fondo de ojo, para luego realizar la clasificación. Usaron entre 10.000 a 1.250.000 imágenes, dependiendo de la lesión, para el entrenamiento de las redes, obteniendo un área bajo la curva ROC de 0.98, una medida de qué tan bueno es el clasificador.

Otros resultados interesantes en este ámbito fueron los obtenidos en la competencia de Kaggle: *Diabetic Retinopathy Detection*. En esta competencia el data set utilizado fue de más de 85.000 imágenes de fondo de ojo, y la tarea a resolver fue la clasificación de las imágenes en las 5 categorías correspondientes con los niveles de severidad. La solución del grupo ganador consistió de una combinación de 3 modelos distintos de redes convolucionales, cada una con un tamaño de imagen de entrada diferente, combinando los tres modelos usando un algoritmo de random forest. Obtuvieron una precisión de un indice kappa pesado cuadráticamente de 0.85. Este índice es una medida de la concordancia de la salida de la red con la clase esperada.

Otro trabajo importante fue el hecho por Gulshan et. al [12], en el cual diseñan y validan una red neuronal convolucional profunda para la detección de retinopatía diabética. En el trabajo utilizan la arquitectura de la red Inception V3, entrenando 10 redes de este tipo, entrenando todos los pesos de las mismas usando más de 128.000 imágenes, llegando un área bajo la curva ROC de 0.99.

Finalmente el último trabajo citado, es el realizado por Pires et al [13], en el cual se entrena una red convolucional con una arquitectura propuesta por su grupo, entrenando con distintos tamaños de imágenes y utilizando 3 datasets, con un total de cerca de 90.000 imágenes.

Como se observa, todos los set de datos utilizados son mucho más grandes que el utilizado en el presente trabajo, que consiste de 5000 imágenes, por lo que es realmente interesante explorar los resultados que se pueden

obtener con tan poca cantidad de datos, usando transferencia de aprendizaje. También es de gran interés analizar las soluciones obtenidas, ya que la confiabilidad de los resultados listados anteriormente se basa completamente en la evaluación de la salida de la red y su concordancia con las categorías esperadas.

Capítulo 3

Método

3.1. Planteamiento del problema

Se trató el problema de clasificación de imágenes de retina para detectar la presencia de retinopatía diabética utilizando transferencia de aprendizaje, con el interés de evaluar el desempeño de dicha técnica en el rango de muy pocas imágenes de entrenamiento, del orden de 5.000.

Se utilizó la base de datos de EyePACS, la cual consiste en 35.000 imágenes de fondo de ojo, cada una de las cuales pertenece a una de las 5 categorías: sin la enfermedad, leve, moderada, severa y proliferativa, con los labels respectivos 0, 1, 2, 3 y 4. De las 35.000, sin embargo, se tomaron 5.000 de esas imágenes. Las imágenes de retina provistas por la base de datos son de alta resolución, con un tamaño promedio de 4752x3168 píxeles, y en la mayoría de los casos se tienen las imagénes de ambos ojos de un mismo paciente.

Una característica de esta base de datos es el gran desbalance entre la cantidad de imágenes en cada categoría, como se puede observar en la figura 3.1. Además, esta base de datos posee una gran cantidad de imágenes de mala calidad, siendo muchas de ellas oscuras, desenfocadas, con mucho contraste

o poco, o con artefactos, mostrando algunos ejemplos en la figura 3.2. Por lo anterior, antes de realizar cualquier tipo de preprocesamiento fue necesario descartar este tipo de imágenes.

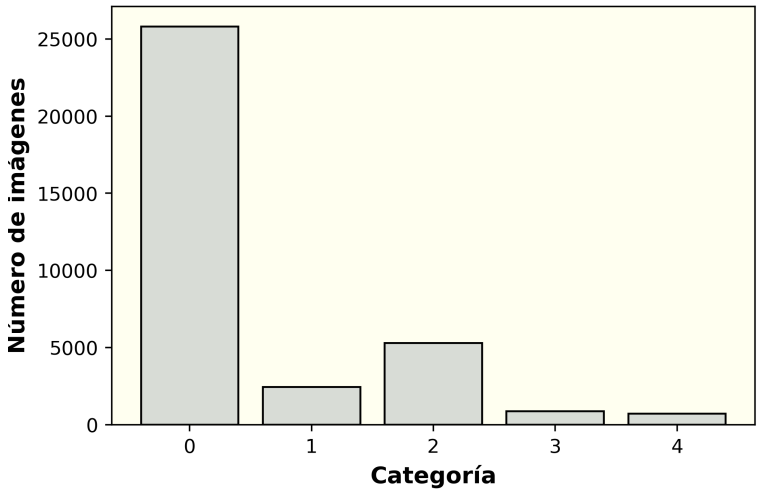
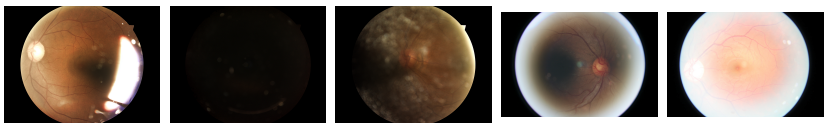


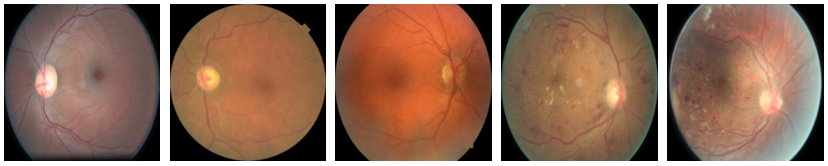
Figura 3.1: Histograma de la cantidad de imágenes por categorías.



(a) (b) (c) (d) (e)

Figura 3.2: Imágenes de la base de datos de EyePACS de mala calidad.

El preprocesamiento inicial de las imágenes consistió en cortarlas, de manera que las imágenes solo contuvieran la retina, para posteriormente realizar un cambio de tamaño. Para lo anterior se escribió un algoritmo que, en función de la intensidad, realizaba el recorte de la imágen y luego el cambio de tamaño. Se utilizaron imágenes de dos tamaños, primero se entrenaron redes con imágenes de 224x224 pixeles y luego redes con imágenes de 448x448 pixeles. Un ejemplo de las imágenes utilizadas como entrada a las redes muestran en la figura 3.3.



(a) (b) (c) (d) (e)

Figura 3.3: Imágenes preprocesadas de las categorías: a) 0 (sin la enfermedad) b) 1 (leve), c) 2 (moderada), d) 3 (severa) y e) 4 (proliferativa).

En lugar de realizar la clasificación en las 5 categorías, se agruparon las imágenes en 2 categorías. Esto se hizo de esta manera siguiendo el interés de que el punto de corte fuese la necesidad de ver a un especialista para decidir el tratamiento apropiado. Normalmente, un paciente cuyo nivel de severidad se encuentra en las categorías 0 o 1, cada 6 meses debe hacer un control para una nueva determinación del estado de su enfermedad. A partir de la categoría 2, el paciente necesita ser evaluado indefectiblemente por un especialista en retina para determinar el procedimiento a seguir y/o el tratamiento a aplicar. Por lo tanto, se realizará una clasificación binaria, con los labels 0 y 1 del nivel de severidad según la ICRP correspondiendo, en nuestra clasificación binaria, a la categoría 0, y los labels 2, 3 y 4 correspondiéndose a nuestra categoría 1.

Las redes preentrenadas utilizadas son: Xception, Inception V3 y DenseNet 169, las cuales tenían pesos provenientes del entrenamiento para resolver la tarea de clasificación de las 1000 categorías de la base de datos ImageNet. Dicha base de datos tiene 14 millones de imágenes.

Cabe destacar que, se encontrará que el primer problema atacado es una tarea más simple que la planteada anteriormente: se entrenaron inicialemente los modelos para realizar una clasificación binaria de las imágenes los niveles 0 y 4 según el ICDR, correspondientes a pacientes sanos o con el nivel más severo de la enfermedad respectivamente, sin tener en cuenta los niveles intermedios. Esta es una instancia exploratoria, por lo que se utilizará un dataset reducido de 1400 imágenes, 700 correspondientes a la categoría 0 y 700

correspondientes a la categoría 4. Se entrenarán primero modelos de clasificación de tipo KNN, SVM, MLP y Random Forest con distintos números de parámetros. Posteriormente se emplean los modelos de redes preentrenadas utilizando la técnica de transferencia de aprendizaje, utilizando los métodos de regularización L2, dropout y el método de data augmentation, todos con el fin de mejorar la generalización de los modelos entrenados.

La primera red entrenada será la red Xception, realizándose de manera exploratoria con el dataset de 1400 imágenes, teniendo de esta manera una noción de los valores de hiperparámetros óptimos. Posteriormente se incorporarán los demás niveles de severidad, manteniendo las dos categorías y se entrenarán las 3 redes, Xception, DenseNet y Inception V3 con el dataset completo de 5000 imágenes, usando 1800 como set de testeo. Se realiza mediante un grid search la búsqueda de hiperparámetros óptimos para cada red.

Para realizar la transferencia, para cada red se tomaron los pesos aprendidos por la parte de la red encargada de la extracción de features o características, correspondiendo con los bloques convolucionales. Se reemplazó, por lo tanto, la última capa densa, que tenía 1000 unidades y era la encargada de la clasificación en del dataset de ImageNet, por dos capas, de 128 y 2 unidades respectivamente para las redes de Xception y DenseNet, y de 256 y 2 unidades para la red Inception V3.

Al realizar el entrenamiento y se tienen dos opciones: la primera opción es congelar los pesos de las capas convolucionales, entrenando solo los pesos de las capas densas. Con este método se dice que la red preentrenada se utiliza como feature extractor. La segunda opción es entrenar, además de los pesos de las capas densas, parte de los pesos de las capas convolucionales provenientes del entrenamiento anterior. En nuestro caso se probaron ambas opciones en un principio y se optó por la segunda técnica, ya que con esta se obtuvieron mejores resultados. A esta última técnica de reentrenamiento se la conoce como *fine tunning*.

En las dos opciones anteriores, Keras permite tener imágenes de entrada de distintos tamaños, con un límite inferior en los mismos de manera de obtener una salida al pasar por todas las capas convolucionales preentrenadas. Como se le agregan una o más capas densas para realizar la nueva tarea de clasificación y se posteriormente se reentrenan dichos pesos, la salida de las capas convolucionales pueden tener un tamaño distinto, entrenándose tantos pesos a la salida como sean necesarios para acomodarse a este nuevo tamaño y a la nueva cantidad de clases correspondientes.

Finalmente se calcula la curva ROC para el problema de clasificación binaria con los 5 niveles de severidad usando otro test set, obtenido de la competencia *APTOS 2019 Blindness Detection*, el cual consta de 600 imágenes.

En cuando a software, se utilizaron las librerías Keras y PyTorch para el entrenamiendo de las redes, teniendo además en uno de sus módulos las redes preentrenadas utilizadas. Para el entrenamiendo se utilizaron GPUs Nvidia GeForce GTX 1080 TI. Cada red preentrenada tiene un tipo de preprocesamiento específico para las imágenes que recibe como entrada y el mismo se encuentra en los módulos usados de Keras o PyTorch.

Capítulo 4

Resultados

4.1. Clasificación binaria: dos niveles de severidad

Se probaron en un principio algoritmos más simples para la clasificación, con imágenes solo de niveles 0 y 4 de severidad, es decir, imágenes de fondo de ojo de pacientes sanos y de pacientes con los signos más severos de la enfermedad. En esta instancia, en lugar de emplear todo el dataset, se utilizaron solo 1400 imágenes del mismo, 700 imágenes de la categoría 0 y 700 de la categoría 4, utilizando la mitad como set de validación y la mitad como set de entrenamiento. Se utilizaron distintos algoritmos para su clasificación: random forest para distinto número de variables de entrada al árbol, KNN con 1, 3, 5 y 7 vecinos, SVM y un perceptrón multicapa o MLP, con una arquitectura de una capa interna de 100 neuronas. Los valores de precisión obtenidos son mostrados en la tabla 4.1, estando todos cercanos a la predicción aleatoria.

Tabla 4.1: Valores de precisión obtenidos utilizando algoritmos de Random Forest, KNN, SVM y MLP para distintos números de parámetros.

Algoritmo		**Precisión**
Random Forest	n=5	0.509
	n=10	0.528
	n=20	0.519
KNN	n=1	0.522
	n=3	0.464
	n=5	0.478
	n=7	0.451
SVM		0.477
MLP		0.522

En vista de que estamos buscando modelos que detecten patrones en las imágenes, particularmente la presencia o ausencia de lesiones, las redes convolucionales son adecuadas para este propósito. Como se comentó anteriormente, su estructura encierra ciertas propiedades e invarianzas convenientes para resolver este tipo de problemas, por lo que nos enfocamos ahora en dichos modelos.

En esta instancia todavía se continuó con la clasificación binaria de imágenes de solo los niveles de severidad 0 y 4, ya que este problema es más sencillo por ser las categorías con la ausencia y la presencia de los signos más severos de la enfermedad respectivamente.

Para este entrenamiento se continuaron utilizando las 1400 imágenes de fondo de ojo de las categorías 0 y 4 empleadas en los métodos anteriores, tomando la mitad como set de entrenamiento y la otra mitad como set de validación, buscando realizar una optimización de hiperparámetros.

Primero se buscó la tasa de aprendizaje y el momento apropiado para el entrenamiento de la red Xception, utilizando SGD como optimizador. Para

ello se hizo un grid search manual, variando la tasa de aprendizaje 4 órdenes de magnitud y 3 valores de momento: 0.9, 0.95 y 0.98, encontrando que los valores de 0.01 para la tasa de aprendizaje y 0.98 para el momento son apropiados para esta red. Se observa que la tasa de aprendizaje es baja, permitiendo un fine tunning de la red. En la figura 4.1 se muestra el entrenamiento de la red Xception para distintas tasas de aprendizaje, con o sin regularización.

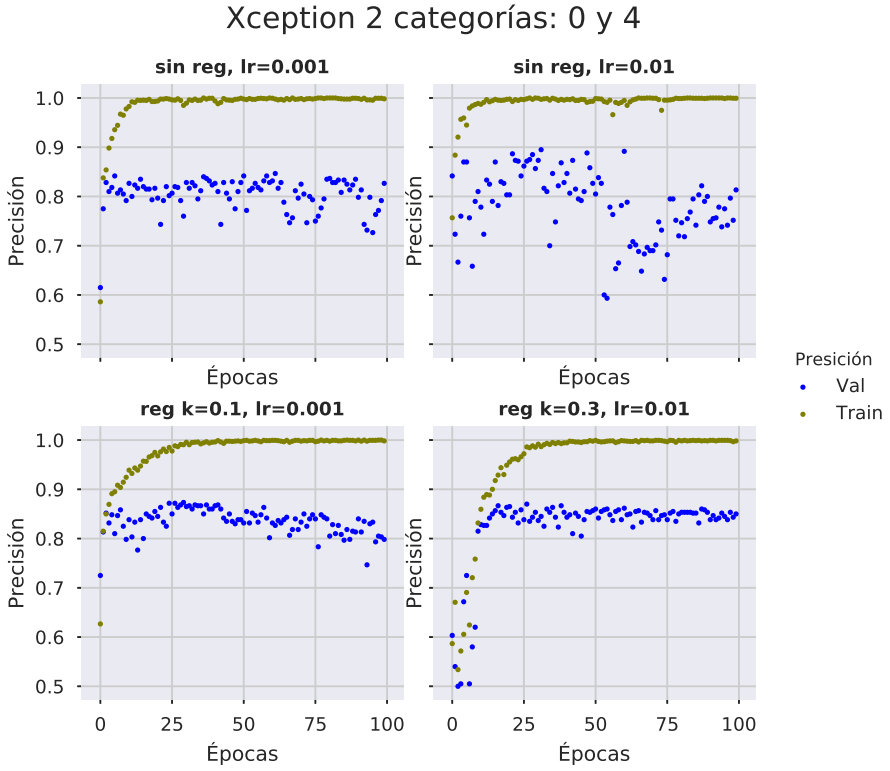


Figura 4.1: Precisión en función de las épocas para el entrenamiento de la red Xception aplicando transferencia de aprendizaje, sin data augmentation y hasta la capa -21. Para todos los entrenamientos se usó SGD como optimizador de la tasa de aprendizaje con un momento de 0.98.

Se observa que en los casos que no se aplica ningún tipo de regularización, el overfitting ocurre más rápido, llegando la precisión del set de entrenamiento a valores cercanos a 1 para épocas más tempranas. Se encontró además un

valor límite de 0.4 para la constante de regularización, valor para el cual la red dejaba de aprender.

En la figura 4.2 se muestran entrenamientos con y sin regularización, con y sin la aplicación de la técnica de dropout. Se observa que si no se aplica ninguna de las dos estrategias, el aprendizaje es más errático y la precisión a la que se llega es menor.

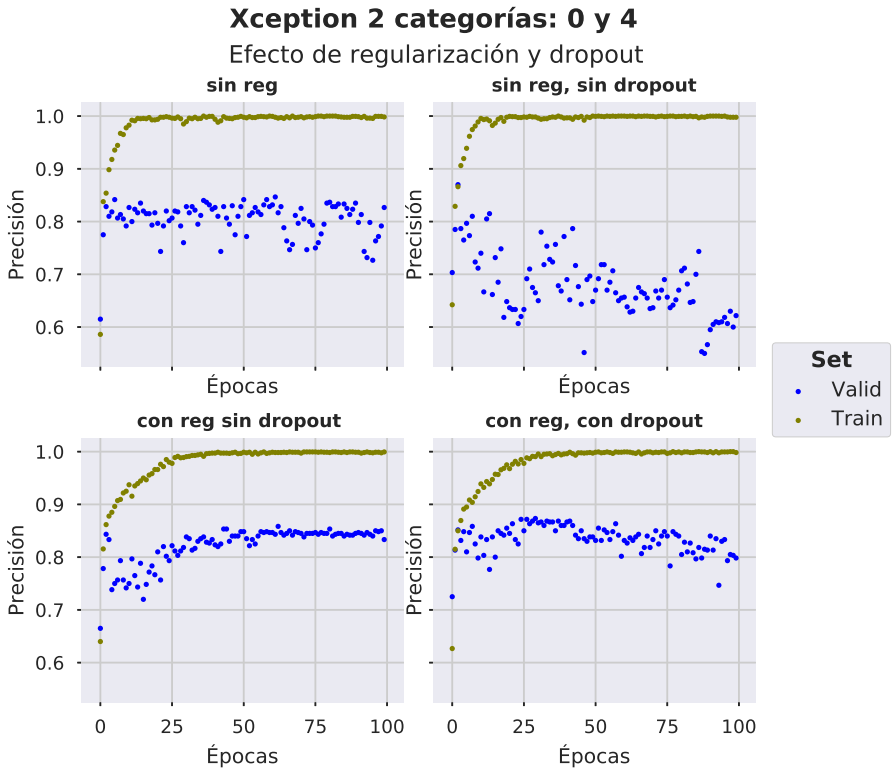


Figura 4.2: Precisión en función de las épocas para el entrenamiento de la red Xception hasta la capa -21, evaluando el funcionamiento de los métodos de dropout y regularización L2.

Al regularizar, se tiene mejor convergencia y al aplicar además dropout, se llega en este caso a mayor precisión con menor cantidad de épocas.

Una cuestión a tener en consideración cuando se realiza transferencia de aprendizaje con fine tunning, es la profundidad en capas hasta la cual se

reentrena, es decir, hasta que profundidad se deben descongelarse los pesos preentrenados. Lo anterior esta íntimamente relacionado con la arquitectura de la red utilizada. Las arquitecturas de las redes preentrenadas utilizadas, como se observó, se encuentran divididas en bloques, donde particularmente para la red Xception los últimos bloques tienen la forma mostrada en la figura 4.3.

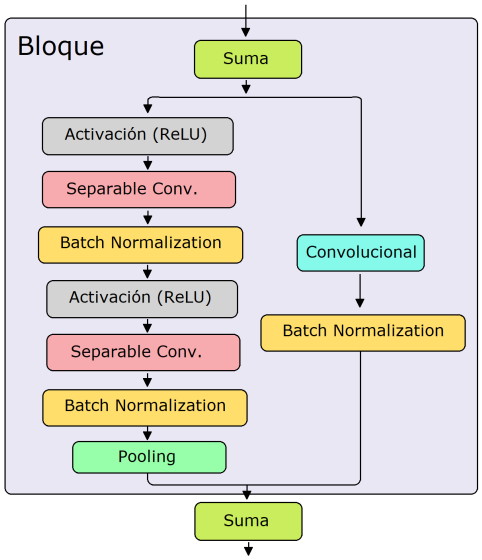


Figura 4.3: Secuencia de capas de los últimos bloques de la red Xception.

Se buscó por lo tanto la relación entre estos bloques con la profundidad del fine tuning, es decir, de qué forma esta estructura en bloques influenciaba hasta donde se debían descongelar los pesos para tener un buen entrenamiento. Se encontró que es conveniente entrenar bloques completos, descongelando pesos de a bloques enteros. Esto se evidencia en la figura 4.4, donde el primer bloque termina en la capa -11 y el segundo en la capa -21, obteniendo para estos dos casos las mayores precisiones y un aprendizaje menos errático que al entrenar hasta capas intermedias, indicadas como -8 y -18 en la figura.

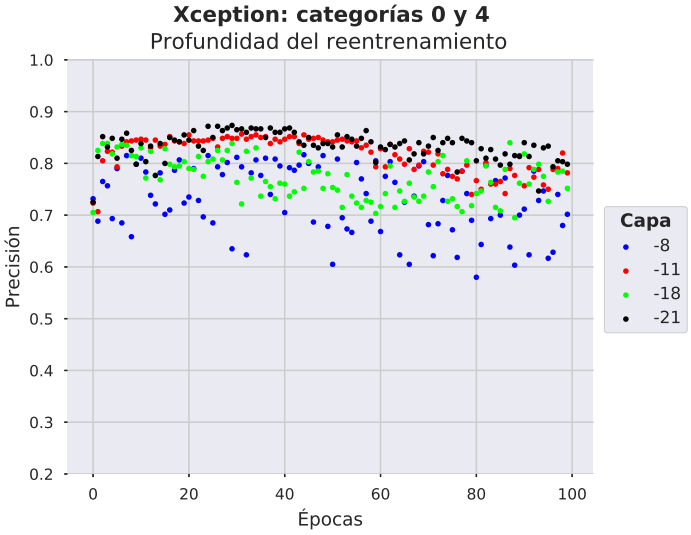


Figura 4.4: Precisión en función de las épocas para el entrenamiento de la red Xception con dos categorías, entrenando hasta las capas -8, -11, -15 y -21.

Una vez que se observó un aprendizaje satisfactorio utilizando solo imágenes de los dos niveles de severidad extremos (sin enfermedad y con los signos severos de la misma), se procedió a incorporar las imágenes de los grados intermedios restantes de la enfermedad y a utilizar, además de la red Xception, las redes DenseNet 169 e Inception V3.

4.2. Clasificación binaria: todos los niveles de severidad

A partir de esta sección se realiza la clasificación binaria de las imágenes divididas en las categorías: 0 para la imágenes correspondientes a los niveles 0 y 1 de severidad, y la categoría 1 correspondiente a imágenes con los niveles 2, 3 y 4 de severidad. Las imágenes utilizadas en esta instancia son de 224x224 pixeles.

Se utilizan en esta instancia, por lo tanto, además de la red Xception, las redes DenseNet169 y Inception V3, preentrenadas todas con la base de datos de ImageNet, particularmente con la base provista por la competencia Large Scale Visual Recognition Challenge (ILSVRC), la cual consta de alrededor de 14 millones de imágenes de entrenamiento para 1000 categorías. Las principales características de cada red se muestran en la tabla 4.2.

Tabla 4.2: Principales características de las redes preentrenadas utilizadas.

Característica	**Xception**	**DenseNet169**	**InceptionV3**
Parámetros	22.910.480	14.307.880	23.851.784
Capas	126	169	159
Precisión Top-1 en ImageNet	0.790	0.762	0.779

Al realizar el entrenamiento de las tres redes con estas nuevas categorías, se realizó la búsqueda de la tasa de aprendizaje y el momento de cada una mediante un grid search, variando entre el momento y la tasa de aprendizaje entre los mismos rangos variados anteriormente, haciendo esto para todas las redes. Para la red Xception se encontró que el valor de tasa de aprendizaje apropiado es de 0.001, 10 veces menor que el valor utilizado en el problema de clasificación de la sección anterior, el cual es más sencillo, indicando esto que al ser más complicado el problema, la velocidad apropiada a la que se deben modificar los pesos debe ser más lenta. Para la red Inception V3 se encontró que el valor de learning rate apropiado es de 0.0001 y el momento de 0.98, mientras que para la red DenseNet 169 la tasa de aprendizaje apropiada es de 0.001 y el momento es de 0.98.

Para aumentar la precisión obtenida, además de utilizar las técnicas de dropout y de regularización L2 empleadas en la sección anterior, fue necesario utilizar la técnica de data augmetation. Se encontró que la constante de regularización apropiada es 2 órdenes de magnitud menor a la utilizada en el problema de clasificación anterior. Esto tiene relación con que se tiene mayor

cantidad de imágenes, ya que se incorporan los niveles de severidad restantes, por lo que se tiene más features extraíbles por tener más lesiones que detectar. Por lo anterior es más sencillo generalizar y por lo tanto se vuelve menos necesario regularizar, llevando a una constante de regularización más chica.

Se volvió a buscar si había una profundidad a la cual la precisión obtenida fuese mayor, obteniendo los resultados mostrados en la figura 4.5. Se observa que a diferencia del entrenamiento anterior con solo 2 niveles de severidad, existe para este caso una profundidad óptima, la cual es entrenar hasta 3 bloques, lo que corresponde en el gráfico a los datos mostrados como *capa -31*.

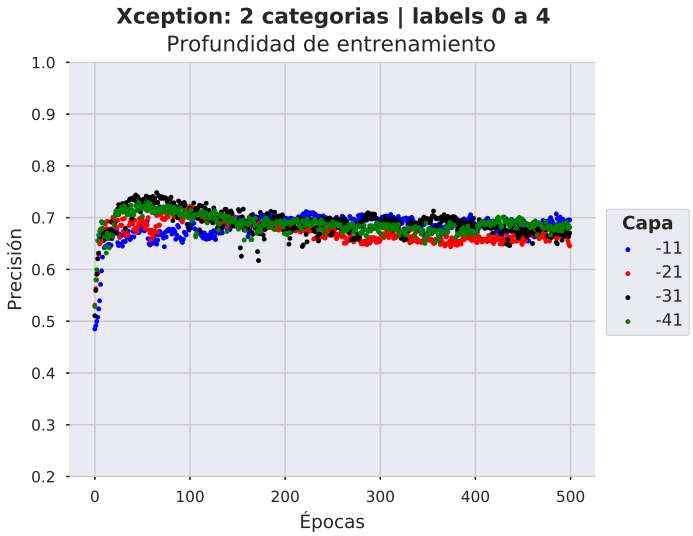


Figura 4.5: Secuencia de capas de los últimos bloques de la red Xception.

El mismo procedimiento se realizó para la red DenseNet 169, encontrando una profundidad apropiada de entrenamiento de 2 bloques y para la red Inception V3 de 1 bloque.

Si bien el problema de clasificación binaria empleando solo 2 niveles de severidad es similar al problema de clasificación binaria usando todos los niveles de severidad, este último resulta más complicado, ya que la cantidad de lesiones que tiene que identificar son mayores. Ahora para clasificar una

imagen como de categoría 0, la red tiene que indentificar la ausencia o la presencia de lesiones leves en la misma. Para las imágenes de la categoría 1 tiene que identificar lesiones moderadas, severas o proliferativas. Además, como las diferencias entre los niveles de severidad 0 y 4 son más evidentes que las diferencias entre los niveles 0, 1 y 2, 3, 4, el problema a resolver es más complicado, teniendo en cuenta además que si ordenamos el dataset en niveles de severidad, el punto de corte ahora resulta entre los niveles 1 y 2.

Las precisiones obtenidas para las distintas redes se muestran en la tabla 4.3, observando que para las 3 redes utilizadas, los valores obtenidos son muy parecidos.

Tabla 4.3: Precisión obtenida con distintos tamaños de imágenes.

Red/Dimensión	**224x224**
Xception	0.76
DenseNet 169	0.77
Inception V3	0.76

Se procede a analizar las soluciones obtenidas utilizando distintos métodos, con el fin de indagar en los criterios aplicados por las redes para la clasificación de las imágenes, buscando además con esta información intuir como mejorar los modelos y obtener por lo tanto precisiones más altas.

4.2.1. Análisis de las soluciones encontradas

Para empezar se visualizó las salidas obtenidas del set de testeo en la anteúltima capa densa de las soluciones encontradas. Las matrices obtenidas son mostradas en la figuras 4.6, 4.7 y 4.8, teniendo en el caso de las redes Xception y DenseNet169 128 valores de salida y para la red Inception V3 256, correspondientes con el eje x de las imágenes.

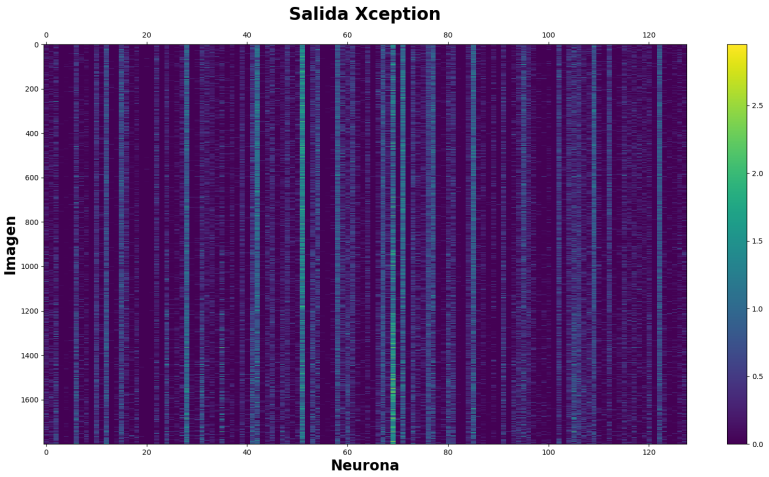


Figura 4.6: Salida de la anteúltima capa de la red Xception.

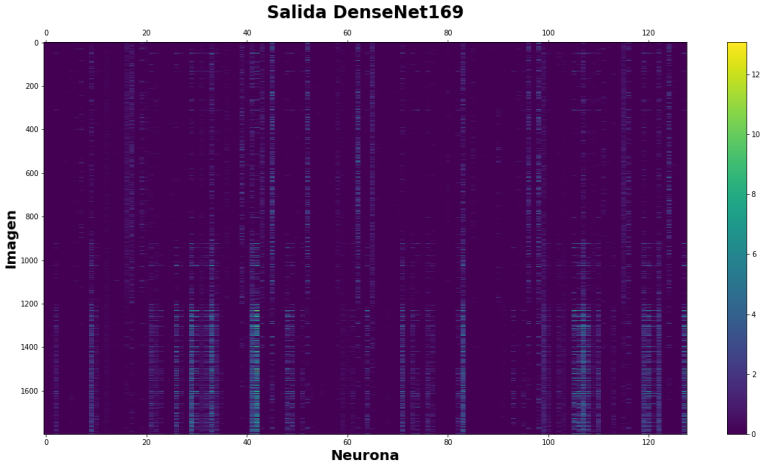


Figura 4.7: Salida de la anteúltima capa de la red DenseNet169.

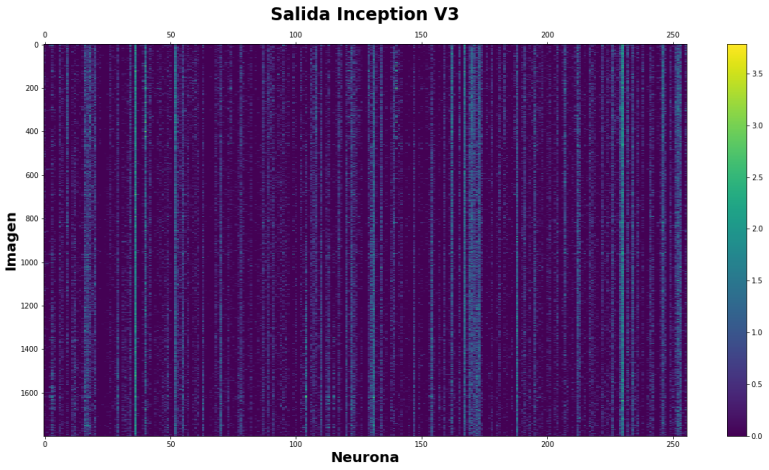
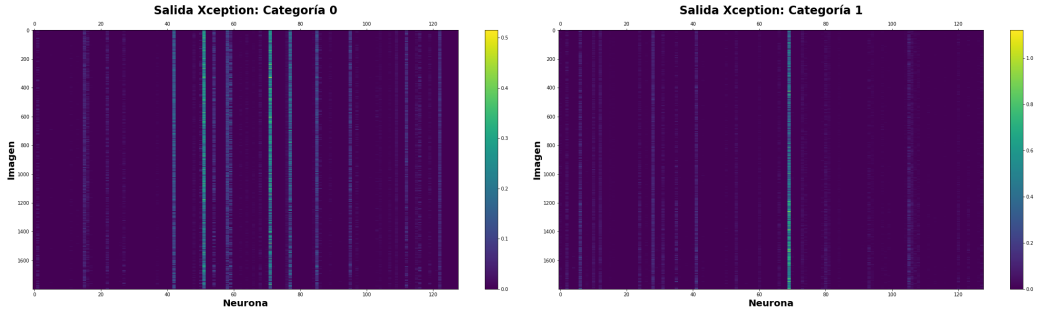


Figura 4.8: Salida de la anteúltima capa de la red InceptionV3.

Es importante destacar que las salidas se encuentran ordenadas por nivel de severidad, de arriba hacia abajo, es decir, los valores de la parte superior corresponden a las imágenes de un nivel de severidad 0 (sano), luego siguen las del nivel 1 (leve) y así sucesivamente.

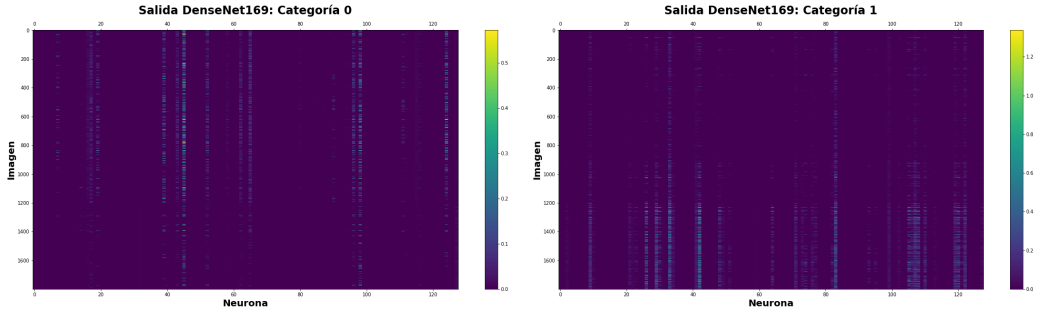
Observando las imágenes obtenidas, no se distinguen a simple vista diferencias importantes en el patrón de salida que llevaría a la red a distinguir una categoría de la otra, excepto un leve patrón en los valores obtenidos para la red DenseNet 169, en el cual ciertas salidas aumentan de valor con la severidad a la que pertenece la imagen.

En búsqueda de un patrón más claro, se multiplicaron las matrices por los pesos correspondientes para obtener las entradas a las unidades que clasifican cada categoría, obteniendo los resultados mostrados en las figuras 4.9, 4.10 y 4.11.



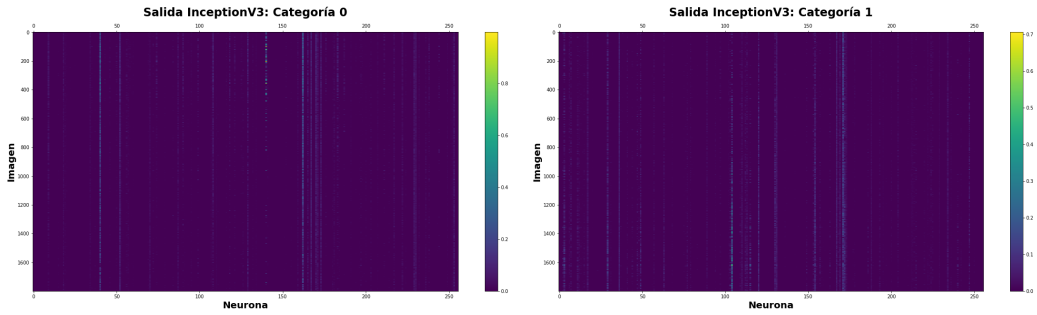
(a) (b)

Figura 4.9: Salida de la anteúltima capa de la red Xception importantes para las categorías 0 y 1.



(a) (b)

Figura 4.10: Salida de la anteúltima capa de la red DenseNet169 importantes para las categorías 0 y 1.



(a) (b)

Figura 4.11: Salida de la anteúltima capa de la red Inception V3 importantes para las categorías 0 y 1.

Al ver posteriormente cuáles salidas son importantes para cada categoría, multiplicando estas salidas por los pesos que las conectan con la unidad de la capa de salida correspondiente a la categoría de interés, se puede distinguir mejor la diferencia entre las salidas relevantes a cada categoría. Para la red Xception, sin embargo, dicha distinción todavía no resulta muy clara. Se puede observar además que el aumento gradual de la salida de algunas unidades se corresponde con el orden en severidad de las imágenes, de arriba hacia abajo, como es de esperarse, ya que las lesiones identificables se hacen más evidentes con la severidad de la enfermedad.

De las figuras anteriores se puede observar también que no todas las salidas son utilizadas para realizar la clasificación de las imágenes, y esto tiene relación con el empleo de la técnica de transferencia de aprendizaje. Al estar utilizando gran parte de la red pesos preentrenados, es posible que se detecten features que no son relevantes específicamente para la tarea actual, la detección de lesiones propias de la retinopatía diabética.

En este punto es de interés poder observar las salidas de las imágenes, que son puntos en un espacio 128 o 256-dimensional, en un espacio de 2 dimensiones. De esta manera es posible discernir si hay algún tipo de separabilidad

clara de las salidas o la formación de clusters correspondientes con las clases que la red intenta clasificar. Es de esperar que para esta instancia exista algún tipo de separabilidad evidente, ya que para este punto solo queda la multiplicación por los pesos correspondientes a la clase en cuestión y la evaluación de la salida de la red. Para este propósito, el método de reducción de dimensionalidad t-SNE [14] es una técnica apropiada, ya que es un método de reducción de dimensionalidad no lineal que intenta mantener la vecindad de puntos de alta dimension en un espacio de menor dimensión. Al aplicar el método a los valores de salida se obtuvieron los resultados mostrados en las figuras 4.12, 4.14 y 4.13.

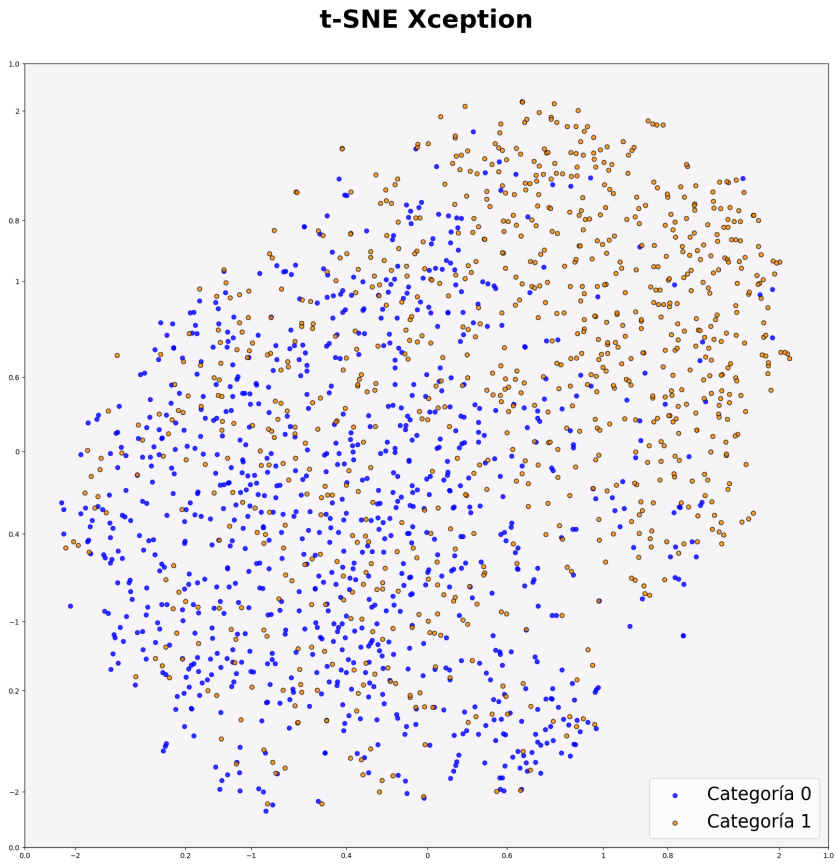


Figura 4.12: t-SNE aplicado a las salidas obtenidas de la anteúltima capa de la red Xception.

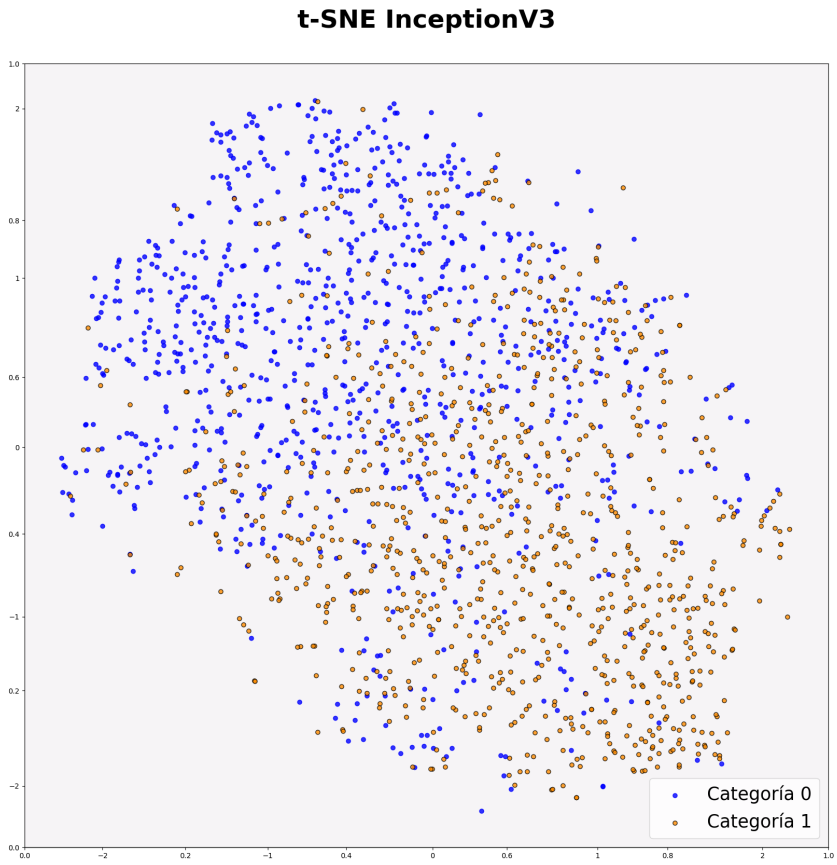


Figura 4.13: t-SNE aplicado a las salidas obtenidas de la anteúltima capa de la red Inception V3.

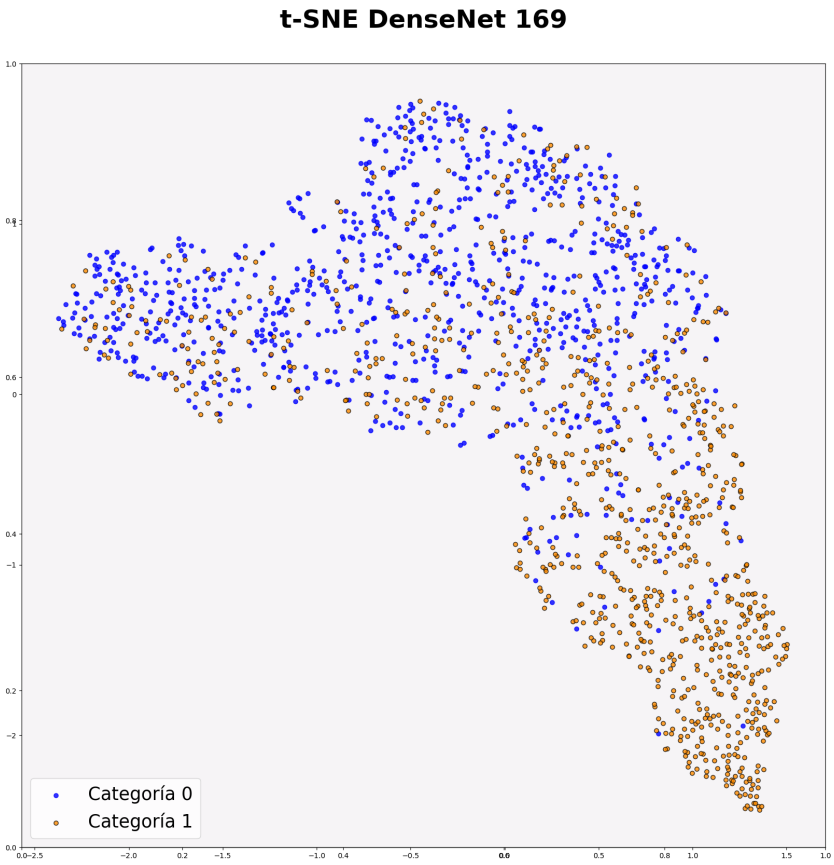


Figura 4.14: t-SNE aplicado a las salidas obtenidas de la anteúltima capa de la red DenseNet 169.

Se observa la formación de clusters con las imágenes de cada categorías, sin embargo, existe un gran solapamiento entre las mismas. Para la salida de la red DenseNet169 se observa una mayor separación, teniendo esto relación con un patrón de salida para cada categoría más distinguible que el correspondiente al de las otras redes.

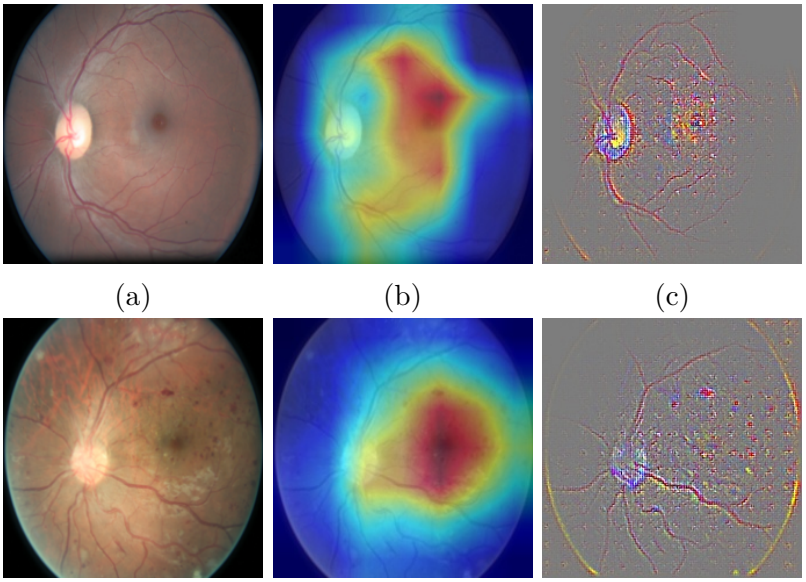
Es interesante preguntarse por qué, a pesar de que las precisiones obtenidas por las 3 redes son parecidas, no se observa igual separabilidad al aplicar el método de reducción de dimensionalidad t-SNE. Como se observó en las salidas obtenidas en las imágenes 4.9, 4.10, y 4.11, para la red Xception es

menos evidente la diferencia entre las salidas correspondientes a cada categoría. Para las salidas de la red DenseNet 169 en cambio, la diferencia es mucho más evidente, teniendo incluso una gradualidad y una correspondencia con el nivel de severidad de la imagen. Por lo tanto, para las salidas cuyo patrón es más evidente, la separabilidad en 2 dimensiones va a ser mayor, ya que los patrones más importantes para la clasificación son los observados en unas pocos valores de la salida, mientras que para la red Xception, la importancia de las salidas para la clasificación se encuentra distribuida más uniformemente. Se obtuvo por ello para la red DensetNet mayor separabilidad al aplicar t-SNE y menor para la red Xception.

Es evidente por lo tanto que al aplicar reducción de dimensionalidad, 2 dimensiones no son suficientes para realizar una clasificación con una precisión como la obtenida por la red, especialmente para la red Xception, y es por lo tanto una combinación de las salidas de esta capa la que permite llegar a dichas precisiones.

Mapas de activación de clases

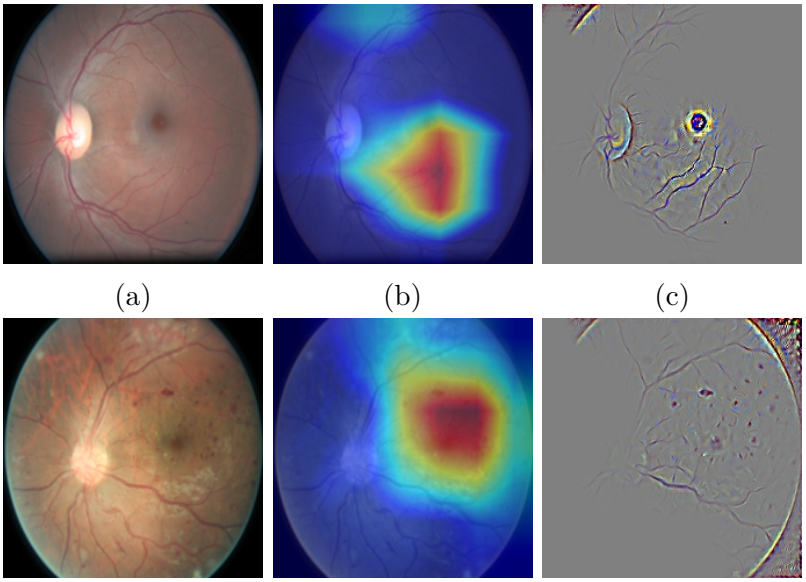
Se aplicaron los métodos CAM y guided GradCAM a las redes entrenadas para observar las regiones de las imágenes que tienen más importancia a la hora de realizar la clasificación por la red. Los resultados obtenidos se muestran en las figuras 4.15,4.16 y 4.17.



(a) (b) (c)

(d) (e) (f)

Figura 4.15: a) Imagen de la categoría 0. b) Imagen de la categoría 1. b) y d) Aplicación de los métodos CAM a las imágenes a) y b) respectivamente. c) y f) aplicación del método Guided GradCAM a las imágenes a) y b) respectivamente. Métodos aplicados a la solución obtenida con la red Xception con imágenes de tamaño 224x224.



(a) (b) (c)

(d) (e) (f)

Figura 4.16: a) Imagen de la categoría 0. b) Imagen de la categoría 1. b) y d) Aplicación de los métodos CAM a las imágenes a) y b) respectivamente. c) y f) aplicación del método Guided GradCAM a las imágenes a) y b) respectivamente. Métodos aplicados a la solución obtenida con la red DenseNet con imágenes de tamaño 224x224.

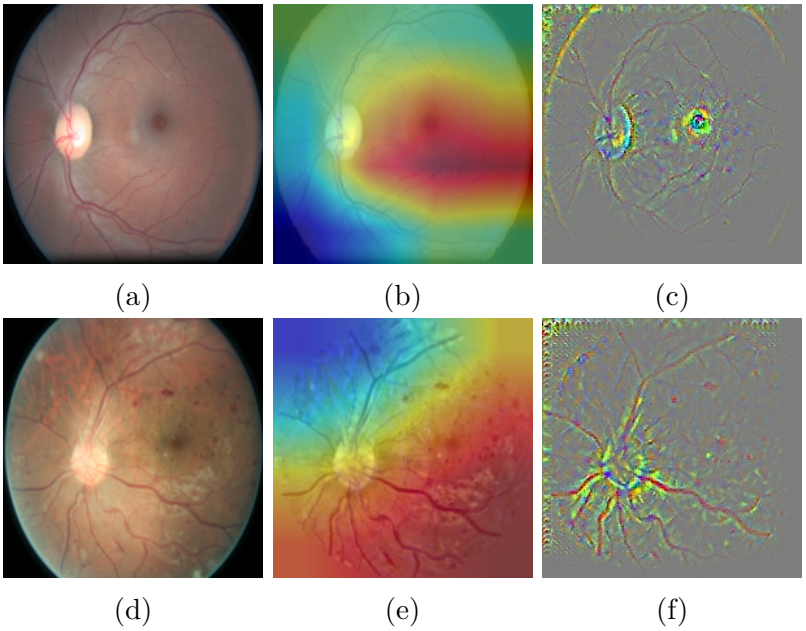


Figura 4.17: a) Imagen de la categoría 0. b) Imagen de la categoría 1. b) y d) Aplicación de los métodos CAM a las imágenes a) y b) respectivamente. c) y f) aplicación del método Guided GradCAM a las imágenes a) y b) respectivamente. Métodos aplicados a la solución obtenida con la red Inception V3 con imágenes de tamaño 224x224.

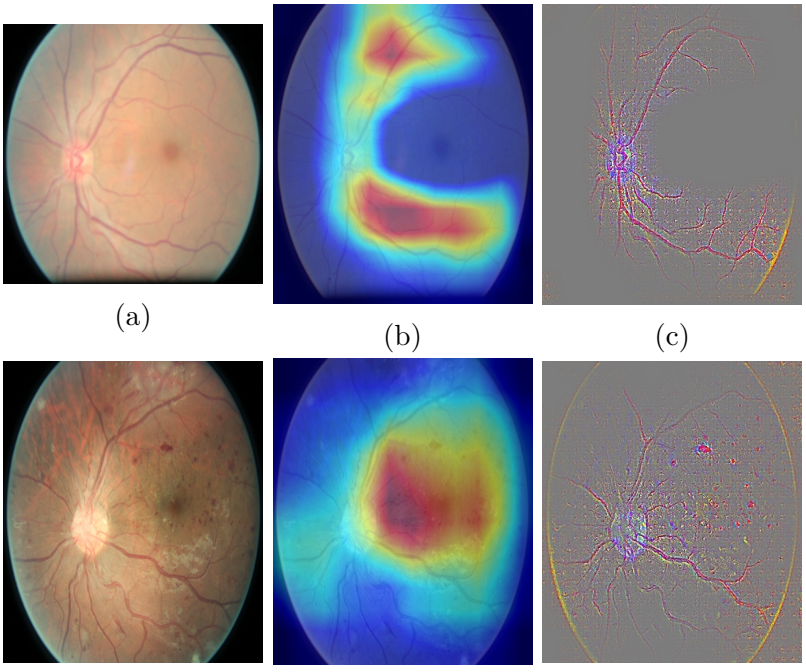
Para todos los casos se observa que la aparición de microhemorragias y exudados tienen importancia a la hora de realizar la tarea de clasificación, lo cual se corresponde con el tipo de identificación que realizan los especialistas a la hora de clasificar la imagen.

Se observa que los pequeños detalles de la imágen son importantes, ya que muchas de las lesiones presentes en las primeras etapas de la patología son de poco tamaño. Siguendo esta línea de pensamiento, se utilizaron imágenes de mayor tamaño para el entrenamiendo, buscando de esta manera una mejora en la precisión en la tarea de clasificación. Particularmente se utilizaron imágenes de un tamaño de 448x448 pixeles, el doble de tamaño que antes. Los resultados de precisión obtenidos son mostrados en la tabla 4.4.

Tabla 4.4: Precisión obtenida con distintos tamaños de imágenes.

Red/Dimensión	**224x224**	**448x448**
Xception	0.76	0.83
DenseNet 169	0.77	0.82
Inception V3	0.76	0.84

Se observa que la precisión aumentó al incrementar el tamaño de las imágenes. Se aplicaron posteriormente los métodos CAM y guided GradCAM la redes entredas con las imágenes de 448x448, obteniendo los resultados mostrados en las figuras 4.18 y 4.19.



(a) (b) (c)

(d) (e) (f)

Figura 4.18: a) Imagen de la categoría 0. b) Imagen de la categoría 1. b) y d) Aplicación de los métodos CAM a las imágenes a) y b) respectivamente. c) y f) aplicación del método Guided GradCAM a las imágenes a) y b) respectivamente. Métodos aplicados a la solución obtenida con la red DenseNet con imágenes de tamaño 448x448.

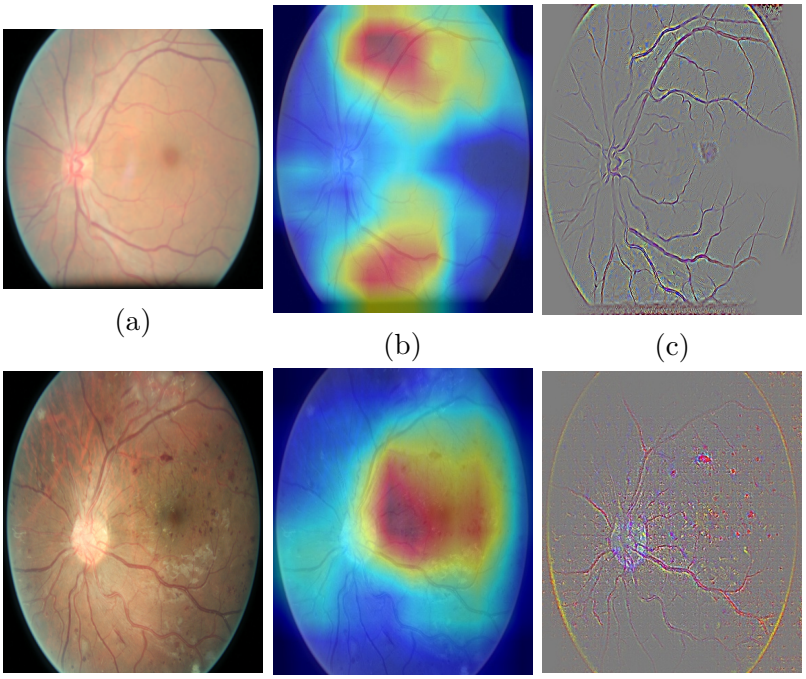
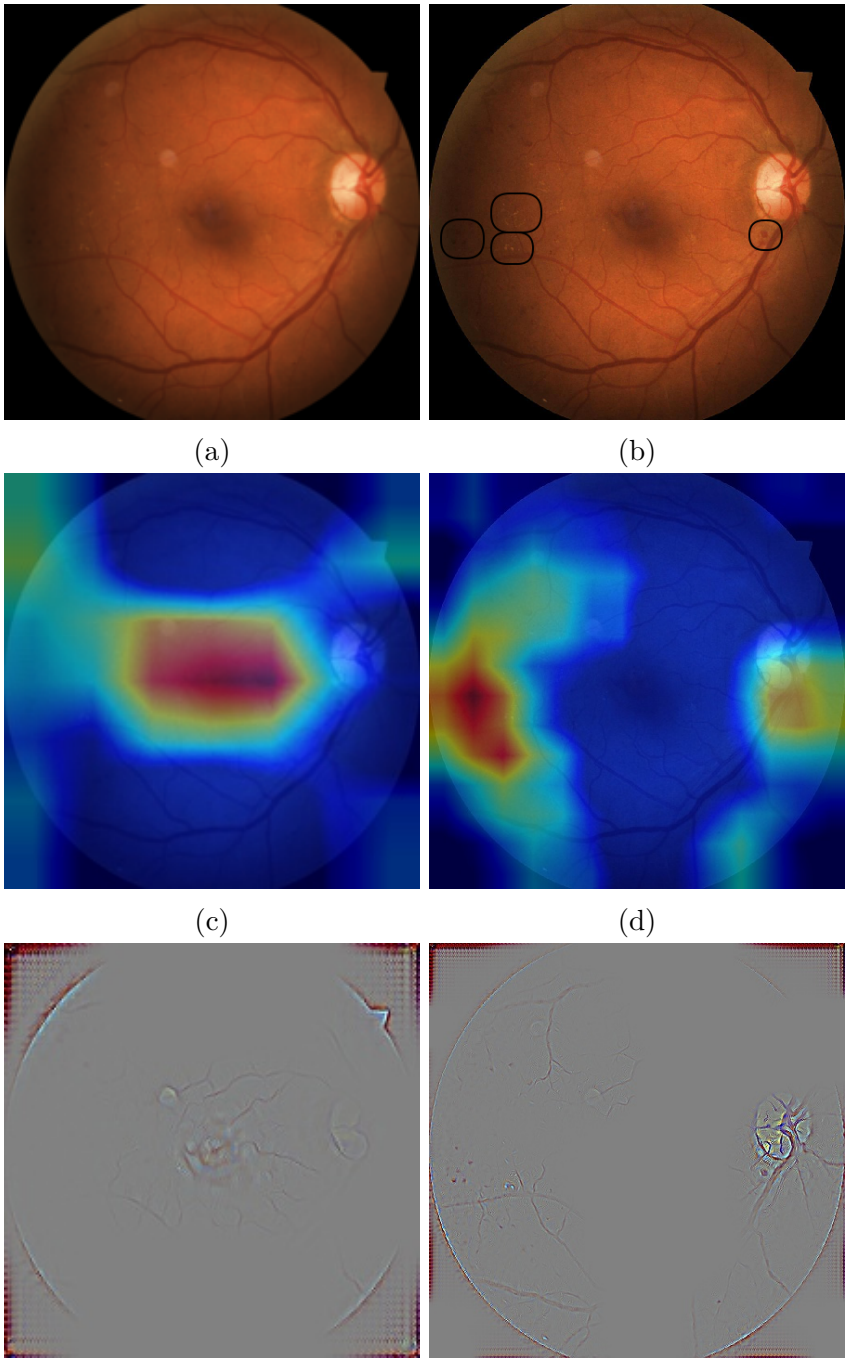


Figura 4.19: a) Imagen de la categoría 0. b) Imagen de la categoría 1. b) y d) Aplicación de los métodos CAM a las imágenes a) y b) respectivamente. c) y f) aplicación del método Guided GradCAM a las imágenes a) y b) respectivamente. Métodos aplicados a la solución obtenida con la red Xception con imágenes de tamaño 448x448.

Se observa que al tener las redes mayor tamaño de imágenes de entrada, las regiones de los mapas de activación de clases son un poco más específicas en cuanto a la zona de importancia, observando además que los pixeles de la imagen resaltados por el métodos guided GradCAM muestran mayor cantidad de detalles, correspondiente con una mayor resolución en la identificación de vasos sanguíneos y lesiones en la retina.

Es interesante indagar en la razón de la mejora en la precisión obtenida y evidenciar la relevancia que tiene el tamaño de la imagen en la tarea de clasificación. Para ello se tomó una imagen que era clasificada erróneamente por la red cuya entrada es de tamaño 224x224, pero que es clasificada correctamente por la red que toma como entrada imágenes de 448x448. En la

figura 4.20 se muestra la aplicación de los métodos CAM y guided GradCAM a dichas redes. Las imágenes 4.20a y 4.20b corresponden a la imágen anterior tomada como entrada para las redes con los tamaños 224x224 y 448x448 respectivamente. La imagen pertenece a la categoría 1, recuadrando las lesiones más visibles en 4.20b, correspondientes a exudados duros y microhemorragias. En la imágen 4.20a, por ser de menor tamaño y por lo tanto tener menos nivel de detalle, las lesiones son menos visibles.



(a) (b) (c) (d) (e) (f)

Figura 4.20: Aplicación de los métodos CAM, Guided GradCAM para una imagen que es clasificada de manera errónea por la red con entradas de tamaño 224x224(izquierda) y clasificada correctamente por la red con entradas de tamaño 448x448.

Es importante aclarar que para ambas redes se buscaron los mapas de activación que máximizan la pertenencia de la imagen a la categoría 1, aún cuando la red con entrada 224x224 la clasifica erróneamente como categoría 0. Esto se realizó para ver si detectaba o no alguna lesión. En las figuras 4.20c y 4.20d se muestran regiones de activación, observando que la razón por la que es clasificada erróneamente es porque la red de 224x224 no detecta de las lesiones, observando además que en la red cuyas imágenes de entrada tenían más detalle, se resaltan dichas lesiones.

Finalmente, recordando que en machine learning el mayor desafío es obtener una buena precisión en un set de datos no visto por la red, se utilizó otro set de testeo para calcular la curva ROC utilizando 600 imágenes de la base de datos de la competencia *APTOS 2019 Blindness Detection* obteniendo los resultados para cada red mostrados en las figuras 4.21, 4.22 y 4.23.

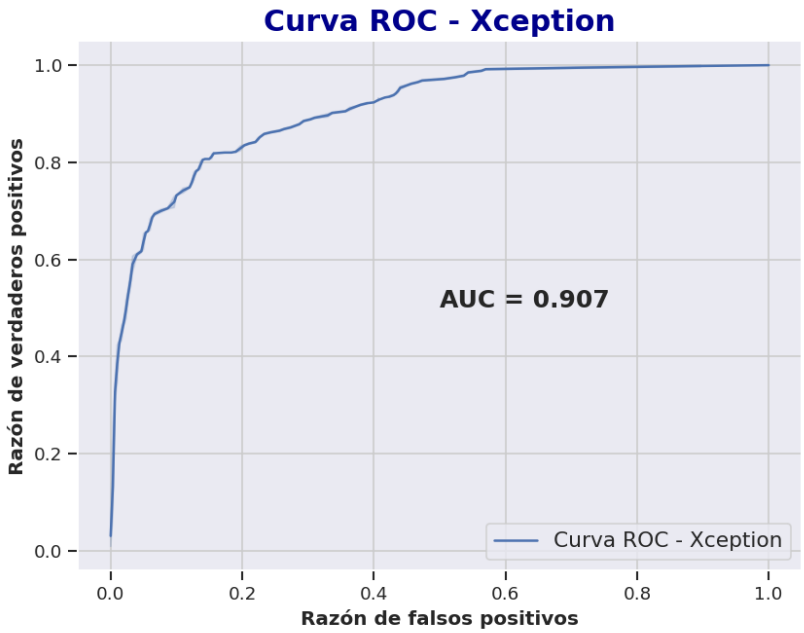


Figura 4.21: Curva ROC para la red Xception.

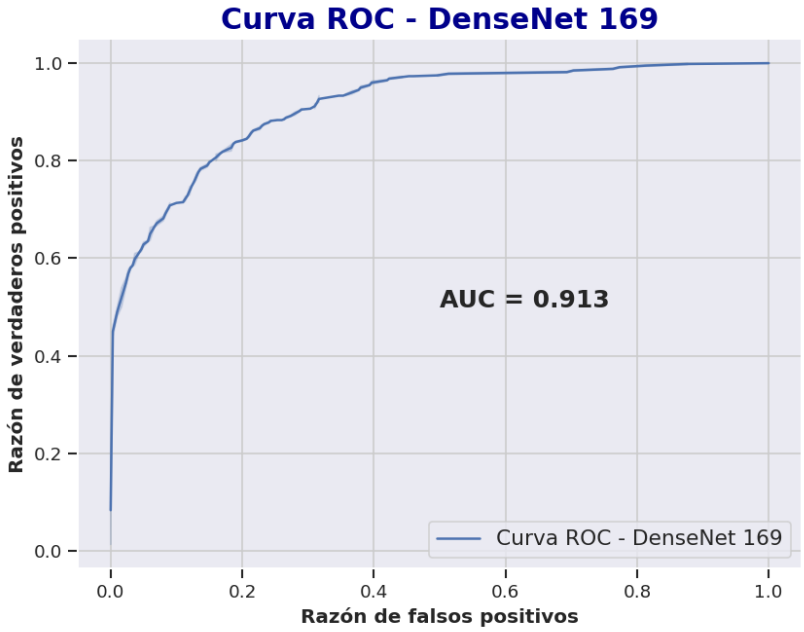


Figura 4.22: Curva ROC para la red DenseNet 169.

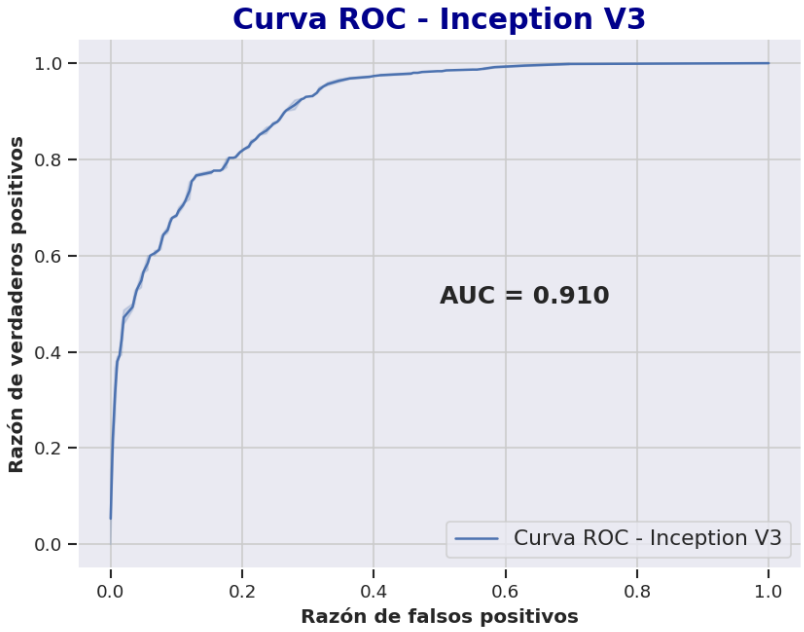


Figura 4.23: Curva ROC para la red Inception V3.

Se observa un buen poder discriminativo para los 3 casos, con un valor de AUC de cerca de 0.9 para todos los casos.

Conclusiones

En el presente trabajo se utilizaron redes neuronales convolucionales profundas para la clasificación de imágenes de fondo de ojo para detectar retinopatía diabética. Se utilizaron distintos métodos para analizar las soluciones obtenidas y la salida de la anteúltima capa. A continuación se mencionan las conclusiones obtenidas.

1. Justificación de uso de redes neuronales

Utilizando clasificadores de tipo KNN, Random Forest, SVN y Perceptrón Multicapa se obtuvieron valores de precisión cercanos a la predicción aleatoria, por lo que se justifica el uso de redes convolucionales para le problema planteado, obteniendo posteriormente con estas últimas valores de precisión que se alejan de la predicción aleatoria.

2. Transferencia de aprendizaje en un dataset de muy poca imágenes

Se logró aplicar la técnica de tranferencia de aprendizaje, utilizando las redes Xception, Inception V3 y DenseNet 169, preentrenadas con la base de datos ImageNet de 14 millones de imágenes. Se utilizaron 5000 imágenes de fondo de ojo para el fine tunning, estando esta cantidad dentro del rango de muy pocas imágenes de entrenamiento.

3. Profundidad del fine tuning

Se encontró que al descongelar los pesos de las redes preentrenadas para realizar fine tuning, es conveniente tener en cuenta la estructura en bloques de las redes, descongelando los pesos de a bloques para tener un aprendizaje menos errático y con mayor precisión final.

4. Papel de las técnicas de regularización:

Se utilizaron los métodos de regularización L2, la técnica de dropout y data augmentation, para mejorar la generalización, siendo importantes a la hora de aplicar transferencia de aprendizaje y sobre todo en nuestro caso, cuando se trabaja con un dataset de muy pocas imágenes.

5. Análisis de valores de salida, reducción de dimensionalidad

Observando la salida de la anteúltima capa, para la red DenseNet 169 se encontró que es distinguible el patrón de la salida correspondiente a cada categoría, mientras que para la red Xception el patrón es menos evidente. Se relacionó esto con una mayor separabilidad de los clusters al aplicar el método de reducción de dimensionalidad t-SNE, concluyendo que al utilizar mayor cantidad de valores de la salida, son necesarias más dimensiones para llegar a una separabilidad en los clusters que permita obtener las precisiones obtenidas.

6. Importancia del tamaño de las imágenes

Se observó que el tamaño de la imagen de entrada es importante, encontrando que para las redes con imágenes de 224x224 algunas lesiones no eran detectadas, mientras que dichas lesiones fueron detectadas por redes de tamaño de imagen de 448x448 pixeles.

7. Métodos de visualización de regiones de importancia, criterios de clasificación

Se utilizaron los métodos CAM y guided Grad CAM, mostrando qué partes de la imagen son relevantes para la red al realizar una clasificación. Se confirmó que el criterio que encontró la red es correcto, detectando las

lesiones propias de la retinopatía diabética en las imágenes. A partir de esto se pudo observar por qué la red de 224x224 clasificaba mal algunas imágenes.

8. Resultados en una base de datos de testeo

Se calculó el área bajo la curva ROC en una base de datos de testeo, obteniendo valores de 0.91. Si bien los valores obtenidos en la literatura son de 0.98, el tamaño de los datasets de imágenes utilizados para el entrenamiento eran 10 o hasta 100 veces mayores que el utilizado en el presente trabajo, por lo que dicho resultado es muy satisfactorio teniendo en cuenta nuestro interés de trabajar con un dataset de muy pocas imágenes.

Índice de figuras

Índice de cuadros

Agradecimientos

Quiero agradecer a todos los que de alguna manera aportaron a la conclusión de este trabajo, en especial a mi director Damián y a mi codirectora Inés, por todos los consejos y todo el apoyo brindado durante este tiempo. Quisiera agradecer al Instituto Balseiro y al Departamento de Física Médica por brindarme todos lo medios necesarios, siendo gran parte de este tiempo como un segundo hogar.

Bibliografía

[1] Ryan Lee, T-Y Wong, and Charumathi Sabanayagam. Epidemiology of diabetic retinopathy, diabetic macular edema and related vision loss. *Eye and Vision*, 2, 12 2015.

[2] Frank F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[3] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[4] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[5] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Julio 2017.

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[8] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[9] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. *CoRR*, abs/1512.04150, 2015.

[10] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[11] Michael David Abràmoff, Yiyue Lou, Ali Erginay, Warren Clarida, Ryan Amelon, James C. Folk, and Meindert Niemeijer. Improved automated detection of diabetic retinopathy on a publicly available dataset through integration of deep learning. *Investigative ophthalmology and visual science*, 57 13:5200–5206, 2016.

[12] Varun Gulshan, Lily Peng, Marc Coram, Martin C. Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, Ramasamy Kim, Rajiv Raman, Philip C. Nelson, Jessica L. Mega, and Dale R. Webster. Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. *JAMA*, 316(22):2402–2410, 12 2016.

[13] Ramon Pires, Sandra Avila, Jacques Wainer, Eduardo Valle, Michael Abramoff, and Anderson Rocha. A data-driven approach to referable diabetic retinopathy detection. *Artificial Intelligence in Medicine*, 03 2019.

[14] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.